

[illegible]

```

LL          IIIIII          SSSSSSSS
LL          IIIIII          SSSSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SSSSSS
LL          II             SSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LLLLLLLLLLLL IIIIIIII          SSSSSSSS
LLLLLLLLLLLL IIIIIIII          SSSSSSSS

```

```
1 0001 0 MODULE DBGCVTDX (IDENT = 'V04-000') =
2 0002 0
3 0003 1 BEGIN
4 0004 1
5 0005 1 *****
6 0006 1 *
7 0007 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
8 0008 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
9 0009 1 * ALL RIGHTS RESERVED.
10 0010 1 *
11 0011 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
12 0012 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
13 0013 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
14 0014 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
15 0015 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
16 0016 1 * TRANSFERRED.
17 0017 1 *
18 0018 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
19 0019 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
20 0020 1 * CORPORATION.
21 0021 1 *
22 0022 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
23 0023 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
24 0024 1 *
25 0025 1 *
26 0026 1 *****
27 0027 1
28 0028 1 WRITTEN BY
29 0029 1 Farokh Morshed 01-09-1981
30 0030 1
31 0031 1 MODIFIED BY:
32 0032 1 * These modifications are to LIB$FIND_CVT_PATH, and were done before
33 0033 1 * debug modifications.
34 0034 1 *
35 0035 1 1-001 - Original. FM1001 01-09-1981
36 0036 1 1-002 - Put in a check for DSC$W_LENGTH to be 1 when class A, or NCA, and
37 0037 1 if class NCA stride must be 1. FM 9-9-81
38 0038 1 1-003 - Put in a new data type, DSC$K_DTYPE_VT. FM 1-DEC-81.
39 0039 1 1-004 - Put in a feature where DST_INFO [D_EN] can be picked up for
40 0040 1 LIB$CVT_DX_DX. FM 2-DEC-81.
41 0041 1
42 0042 1 * These modifications are to LIB$CVT_DX_DX, and were done before
43 0043 1 * debug modifications.
44 0044 1 *
45 0045 1 1-001 - Original. FM1001 01-09-1981
46 0046 1 1-002 - Fix the problem with (SMLINT, LRGINT, DEC) to NBDS having an explicit
47 0047 1 sign when plus should be implied. Also [DEC_NBDS] scaled twice,
48 0048 1 changed it to scale only once. FM 5-NOV-81.
49 0049 1 1-003 - Fix the problem with [K_DEC_NBDS]. The length of CLASS_S_DESC was
50 0050 1 not being reset. FM
51 0051 1 1-004 - Put in a new data type, DSC$K_DTYPE_VT. Cleaned up data type B
52 0052 1 out of NBDS. FM 1-DEC-81.
53 0053 1 1-005 - Fix the bug where destination length is not picked up from DST_INFO.
54 0054 1 FM 2-DEC-81.
55 0055 1 1-006 - Constants which are addressed by things like PACK_ZERO should be
56 0056 1 all longwords.
57 0057 1 1-007 - LIB$ROPRAND was left out of the exception handler. FM 8-FEB-82.
```

```
58 0058 1 1-008 - A couple of missing dots fixed -Q -> G and H.
59 0059 1
60 0060 1 * DEBUG modifications start here.
61 0061 1
62 0062 1 1-001 - Victoria Holt Sept., 1982
63 0063 1 Created module DBGCVTDX. This module includes the two routines
64 0064 1 FIND_CVT_PATH and DBG$CVT_DX_DX (originally LIB$FIN_CVT_PATH
65 0065 1 and LIB$CVT_DX_DX, respectively). Both routines have been
66 0066 1 modified to include additional DEBUG and language specific
67 0067 1 dtypes and classes.
68 0068 1 1-002 - VJH
69 0069 1 Added routine DBG$COVER_DX_DX from DBGEVALOP.
70 0070 1 Modified handler so that it signals errors rather than
71 0071 1 returning a status code.
72 0072 1 1-003 - WC3 Jul-83
73 0073 1 Add support for Absolute Date Time to CVT_DX_DX
74 0074 1 1-004 - WC3 Jul-83
75 0075 1 Fix the decimal text to Octaword conversion
76 0076 1 1-005 - BAB Dec-83
77 0077 1 Added support for scaled binary conversions. To and From.
78 0078 1 1-006 - BAB Jan-84
79 0079 1 Fixed the bug where DEP/QUAD I=8000000000000000 does not work.
80 0080 1 Also DEP/OCTA I=80000000000000000000000000000000.
81 0081 1
82 0082 1
83 0083 1 REQUIRE 'SRC$:DBGPROLOG.REQ';
84 0217 1
85 0218 1 LINKAGE
86 0219 1 JSB_R0 = JSB (REGISTER = 0): PRESERVE (0, 1),
87 0220 1 JSB_R1 = JSB (REGISTER = 0, REGISTER = 1): PRESERVE (0, 1),
88 0221 1 JSB_RETRO_R1 = JSB (REGISTER = 0, REGISTER = 1): PRESERVE (1),
89 0222 1 JSB_R2 = JSB (REGISTER = 0, REGISTER = 1, REGISTER = 2): PRESERVE (0, 1),
90 0223 1 JSB_R3 = JSB (REGISTER = 0, REGISTER = 1, REGISTER = 2, REGISTER = 3): PRESERVE (0, 1),
91 0224 1 JSB_R6 = JSB (REGISTER = 0, REGISTER = 1, REGISTER = 2, REGISTER = 3, REGISTER = 4, REGISTER = 5):
92 0225 1 PRESERVE (0, 1),
93 0226 1 SCOPYR JSB_R6 = JSB (REGISTER = 0, REGISTER = 1, REGISTER = 2): NOPRESERVE (2),
94 0227 1 SCOPY JSB_R6 = JSB (REGISTER = 0, REGISTER = 1): NOPRESERVE (2, 3, 4, 5, 6);
95 0228 1
96 0229 1 FORWARD ROUTINE
97 0230 1 DBG$COVER_DX_DX, ! Accepts value descriptors; calls DBG$CVT_DX_DX.
98 0231 1 COVER_VMSDESC_SETUP, ! Set up vms descriptor
99 0232 1 DBG$CVT_DX_DX: NOVALUE, ! Routine that does any-to-any type conversion.
100 0233 1 CVT_HANDLER, ! Error handler.
101 0234 1 FIND_CVT_PATH; ! Routine to find the conversion path
102 0235 1 ! being done and report any
103 0236 1
104 0237 1 EXTERNAL ROUTINE
105 0238 1 DBG$CVT_ASHR_R1: JSB_R6 NOVALUE,
106 0239 1 DBG$CVT_CMPH_R1: JSB_RETRO_R1,
107 0240 1 DBG$CVT_CVTDH_R1: JSB_R1 NOVALUE,
108 0241 1 DBG$CVT_CVTLB_R1: JSB_R1 NOVALUE,
109 0242 1 DBG$CVT_CVTLH_R1: JSB_R1 NOVALUE,
110 0243 1 DBG$CVT_CVTLW_R1: JSB_R1 NOVALUE,
111 0244 1 DBG$CVT_CVTRD_R1: JSB_R1 NOVALUE,
112 0245 1 DBG$CVT_CVTHD_R1: JSB_R1 NOVALUE,
113 0246 1 DBG$CVT_CVTHF_R1: JSB_R1 NOVALUE,
114 0247 1 DBG$CVT_CVTHG_R1: JSB_R1 NOVALUE,
```



```

: 115 0248 1 DBG$CVT_CVTGH R1: JSB R1 NOVALUE,
: 116 0249 1 DBG$CVT_CVTRHC R1: JSB R1 NOVALUE,
: 117 0250 1 DBG$CVT_CVTRHO R1: JSB R1 NOVALUE,
: 118 0251 1 DBG$CVT_CVTRHQ R1: JSB R1 NOVALUE,
: 119 0252 1 DBG$CVT_CVTROUD R1: JSB R1 NOVALUE,
: 120 0253 1 DBG$CVT_CVTROUH R1: JSB R1 NOVALUE,
: 121 0254 1 DBG$CVT_DIVD2 RT: JSB RT NOVALUE,
: 122 0255 1 DBG$CVT_DIVH2 R1: JSB R1 NOVALUE,
: 123 0256 1 DBG$CVT_DIVP R1: JSB R6 NOVALUE,
: 124 0257 1 DBG$CVT_MULD2 R1: JSB R1 NOVALUE,
: 125 0258 1 DBG$CVT_MULH2 R1: JSB R1 NOVALUE,
: 126 0259 1 DBG$CVT_MULP R1: JSB R6 NOVALUE,
: 127 0260 1 DBG$GET-SET TYPEID,
: 128 0261 1 DBG$INS-ENCODE,
: 129 0262 1 DBG$MAP-DTYPE CLASS,
: 130 0263 1 DBG$PERFORM TYPEID CHECK,
: 131 0264 1 DBG$STA-TYP-SUBRNG: NOVALUE,
: 132 0265 1 DBG$STA-TYP-ATOMIC: NOVALUE,
: 133 0266 1 DBG$STRIP-ZEROES,
: 134 0267 1 FOR$CVT-D-TE,
: 135 0268 1 FOR$CVT-D-TF,
: 136 0269 1 FOR$CVT-G-TE,
: 137 0270 1 FOR$CVT-G-TF,
: 138 0271 1 FOR$CVT-H-TE,
: 139 0272 1 FOR$CVT-H-TF,
: 140 0273 1 DBG$CVT-SCALE-OU-UP BY 10 R1: JSB R0 NOVALUE,
: 141 0274 1 DBG$CVT-SCALE-OU-DOWN BY 10 R1: JSB R0 NOVALUE,
: 142 0275 1 LIB$CVT-SCALE-OU-UP BY 10 R1: JSB R0 NOVALUE,
: 143 0276 1 LIB$CVT-SCALE-OU-DOWN BY 10 R1: JSB R0 NOVALUE,
: 144 0277 1 DBG$CVT-SCALE-OU-UP BY 2 R1: JSB R0 NOVALUE,
: 145 0278 1 DBG$CVT-SCALE-OU-DOWN BY 2 R1: JSB R0 NOVALUE,
: 146 0279 1 LIB$MATCH COND,
: 147 0280 1 LIB$SIG TO RET: NOVALUE,
: 148 0281 1 LIB$SCOPY_R DX6: SCOPYR JSB R6,
: 149 0282 1 LIB$SCOPY_DXDX6: SCOPY_JSBR6,
: 150 0283 1 LIB$STOP: NOVALUE,
: 151 0284 1 MTH$CVT-D-G: NOVALUE,
: 152 0285 1 OT$CVT-L-TI,
: 153 0286 1 OT$CVT-T-D,
: 154 0287 1 OT$CVT-T-G,
: 155 0288 1 OT$CVT-T-H,
: 156 0289 1 SYSSASCTIM,
: 157 0290 1 SYSSBINTIM,
: 158 0291 1
: 159 0292 1 EXTERNAL
: 160 0293 1 LIB$AB_CVTTP-U,
: 161 0294 1 LIB$AB_CVT-O-U,
: 162 0295 1 LIB$AB_CVTTP-O,
: 163 0296 1 LIB$AB_CVT-U-O,
: 164 0297 1 LIB$AB_CVTPT-U,
: 165 0298 1 LIB$AB_CVTPT-O,
: 166 0299 1 LIB$AB_CVTPT-Z,
: 167 0300 1 LIB$AB_CVTTP-Z,
: 168 0301 1 DBG$GL_OPCODE_NAME: REF VECTOR[, BYTE]; ! Used in error messages.
: 169 0302 1
: 170 0303 1 EXTERNAL LITERAL
: 171 0304 1 LIB$STRTRU; ! String truncated.

```

! These are the translation tables  
! used when translating to or from  
! packed decimal.

! String truncated.

172	0305	1	
173	0306	1	BUILTIN
174	0307	1	CVTTP.
175	0308	1	CVTSP.
176	0309	1	CVTLF.
177	0310	1	CVTLD.
178	0311	1	CVTPT.
179	0312	1	CVTPS.
180	0313	1	CMPP.
181	0314	1	CMPO.
182	0315	1	CVTDL.
183	0316	1	CVTHL.
184	0317	1	CVTRDL.
185	0318	1	CVTRFL.
186	0319	1	CVTDF.
187	0320	1	CVTPL.
188	0321	1	CVTLP.
189	0322	1	BICPSW.
190	0323	1	BISPSW.
191	0324	1	TESTBITSC.
192	0325	1	SUBM.
193	0326	1	MOVP.
194	0327	1	
195	0328	1	OWN
196	0329	1	DECIMAL_CONVERT.
197	0330	1	
198	0331	1	
199	0332	1	SAVE_RESULT;

! Tells if this is a packed decimal conversion.  
! Needed so that "conversion error" can be  
! signalled rather than "reserved operand".  
! Used when signalling underflow.

```

201 0333 1 ! Literals.
202 0334 1
203 0335 1 LITERAL
204 0336 1
205 0337 1
206 0338 1
207 0339 1
208 0340 1
209 0341 1
210 0342 1
211 0343 1
212 0344 1
213 0345 1
214 0346 1
215 0347 1
216 0348 1
217 0349 1
218 0350 1
219 0351 1
220 0352 1
221 0353 1
222 0354 1
223 0355 1
224 0356 1
225 0357 1
226 0358 1
227 0359 1
228 0360 1
229 0361 1
230 0362 1
231 0363 1
232 0364 1
233 0365 1
234 0366 1
235 0367 1
236 0368 1
237 0369 1
238 0370 1
239 0371 1
240 0372 1
241 0373 1
242 0374 1
243 0375 1
244 0376 1
245 0377 1
246 0378 1
247 0379 1
248 0380 1
249 0381 1
250 0382 1
251 0383 1
252 0384 1
253 0385 1
254 0386 1
255 0387 1
256 0388 1
257 0389 1

! Some general values:
K_FIRST_LONGWORD = 0,
K_INTERMEDIATE_DATA_LENGTH = 32,
K_OUTPUT_BUFFER_LENGTH = 32,
K_LRGST_WU = 65535,
K_LRGST_LU = 4294967295,
K_LRGST_NEG_L = -2147483648,
K_LRGCLSSUP = DSC$K_CLASS_UBS,
K_SMLCLSSUP = DSC$K_CLASS_S,
K_MAX_DATA_TYPES = 43,
K_MAX_CLASSES = 15,
K_MIN_CLASS = DSC$K_CLASS_S,
K_MAX_CLASS = DSC$K_CLASS_UBS,
K_MAX_CLASS_STA = DSC$K_CLASS_UBA,
K_MIN_DTYPE_STA = DSC$K_DTYPE_V,
K_MAX_DTYPE_STA = DSC$K_DTYPE_SVU,
K_ACTUAL_CLASSES = 7,
K_MSTNEGERR = -7,
K_LRGST_NEG_B = -128,
K_LRGST_NEG_W = -32768,
K_LRGST_B = 127,
K_LRGST_W = 32767,
K_LRGST_BU = 255,
K_LRGST_L = 2147483647,
K_PACK_CU_LEN = 10,

! Status returned by FIND_CVT_PATH.
K_UNSCALAROU = -1,
K_UNSDTYROU = -2,
K_UNSDESROU = -3,
K_UNSDESSTA = -4,
K_UNSCLASTA = -5,
K_UNSDTYSTA = -6,
K_INVNBDS = -7,

K_SUPPORTED = 1,

! These are the values of the members of K_ACTUAL_CLASSES:
! (K_ACTUAL_CLASSES being those classes that not only exist, but are
! actually used by the caller. Any other valid classes not in this
! group are presently treated as an error condition.)
K_STATE1_CLASS_S = DSC$K_CLASS_S,
K_STATE2_CLASS_D = DSC$K_CLASS_D,
K_STATE4_CLASS_A = DSC$K_CLASS_A,

! Position of first longword in buffer.
! Intermediate data buffer length
! Output buffer length
! Largest unsigned word.
! Largest unsigned longword.
! Largest negative longword.
! Largest CLASS supported by routine
! Smallest CLASS supported by routine
! Total number of DATA TYPES in the standard
! Total number of classes supported,
! including the error case 0.
! Smallest class supported.
! Largest class supported.
! Max. class number supported by standard.
! Min. data type number supported by standard.
! Max. data type number supported by standard.
! Total classes that are allowed by the STATES table.
! Most negative error state
! Largest negative signed byte.
! Largest negative signed word.
! Largest positive signed byte.
! Largest positive signed word.
! Largest unsigned byte.
! Largest signed longword.

! Unsupported CLASS by routine.
! Unsupported DATA TYPE by routine.
! Unsupported descriptor by routine.
! Unsupported descriptor by standard.
! Unsupported CLASS by standard.
! Unsupported DTYPE by standard
! Invalid NBDS
! because either array size is larger
! than a WU or it is not a one
! dimensional array.
! This descriptor is supported, and valid.
```

```

258 0390 1 K_STATE9 CLASS SD = DSC$K CLASS SD,
259 0391 1 K_STATE10 CLASS NCA = DSC$K CLASS NCA,
260 0392 1 K_STATE11 CLASS VS = DSC$K CLASS VS,
261 0393 1 K_STATE13 CLASS UBS = DSC$K CLASS UBS,
262 0394 1
263 0395 1
264 0396 1
265 0397 1
266 0398 1
267 0399 1
268 0400 1
269 0401 1
270 0402 1
271 0403 1
272 0404 1
273 0405 1
274 0406 1
275 0407 1
276 0408 1
277 0409 1
278 0410 1
279 0411 1
280 0412 1
281 0413 1
282 0414 1
283 0415 1
284 0416 1
285 0417 1
286 0418 1
287 0419 1
288 0420 1
289 0421 1
290 0422 1
291 0423 1
292 0424 1
293 0425 1
294 0426 1
295 0427 1
296 0428 1
297 0429 1
298 0430 1
299 0431 1
300 0432 1
301 0433 1
302 0434 1
303 0435 1
304 0436 1
305 0437 1
306 0438 1
307 0439 1
308 0440 1
309 0441 1
310 0442 1
311 0443 1
312 0444 1
313 0445 1
314 0446 1

K_STATE9 CLASS SD = DSC$K CLASS SD,
K_STATE10 CLASS NCA = DSC$K CLASS NCA,
K_STATE11 CLASS VS = DSC$K CLASS VS,
K_STATE13 CLASS UBS = DSC$K CLASS UBS,

! These are the intermediate data type groupings. All data types fit into
! one of these groups. The groups can represent either the left or right
! hand side of the conversion index. When combined together (as in
! K_SMLINT_SMLINT - convert small integer to small integer, eg. byte to
! word), they represent the current state.

K_SMLINT = 1,
K_LRGINT = 2,
K_SMLFLT_CMPLX = 3,
K_LRGFLT_CMPLX = 4,
K_DEC = 5,
K_NBDS = 6,

K_TOT_CAT = 6.

! These are the index values to the main CASE statement in DBG$CVT_DX_DX.

K_SMLINT_SMLINT = 1,
K_SMLINT_LRGINT = 2,
K_SMLINT_SMLFLT_CMPLX = 3,
K_SMLINT_LRGFLT_CMPLX = 4,
K_SMLINT_DEC = 5,
K_SMLINT_NBDS = 6,
K_LRGINT_SMLINT = 7,
K_LRGINT_LRGINT = 8,
K_LRGINT_SMLFLT_CMPLX = 9,
K_LRGINT_LRGFLT_CMPLX = 10,
K_LRGINT_DEC = 11,
K_LRGINT_NBDS = 12,
K_SMLFLT_CMPLX_SMLINT = 13,
K_SMLFLT_CMPLX_LRGINT = 14,
K_SMLFLT_CMPLX_SMLFLT_CMPLX = 15,
K_SMLFLT_CMPLX_LRGFLT_CMPLX = 16,
K_SMLFLT_CMPLX_DEC = 17,
K_SMLFLT_CMPLX_NBDS = 18,
K_LRGFLT_CMPLX_SMLINT = 19,
K_LRGFLT_CMPLX_LRGINT = 20,
K_LRGFLT_CMPLX_SMLFLT_CMPLX = 21,
K_LRGFLT_CMPLX_LRGFLT_CMPLX = 22,
K_LRGFLT_CMPLX_DEC = 23,
K_LRGFLT_CMPLX_NBDS = 24,
K_DEC_SMLINT = 25,
K_DEC_LRGINT = 26,
K_DEC_SMLFLT_CMPLX = 27,
K_DEC_LRGFLT_CMPLX = 28,
K_DEC_DEC = 29,
K_DEC_NBDS = 30,
K_NBDS_SMLINT = 31,
K_NBDS_LRGINT = 32,
K_NBDS_SMLFLT_CMPLX = 33,
```

```
: 315      0447 1      K_NBDS_LRGFLTCMPLX = 34,
: 316      0448 1      K_NBDS_DEC = 35,
: 317      0449 1      K_NBDS_NBDS = 36,
: 318      0450 1
: 319      0451 1
: 320      0452 1      ! Length of source and destination info records in bytes.
: 321      0453 1      ! The info records are used to hold information gathered in FIND_CVT_PATH,
: 322      0454 1      ! so that it can be easily recovered and used by DBG$CVT_DX_DX.
: 323      0455 1
: 324      0456 1      K_SRC_INFO_LENGTH = 8,
: 325      0457 1      K_DST_INFO_LENGTH = 8,
: 326      0458 1      K_TEMP_BUF_LENGTH = 50,
: 327      0459 1
: 328      0460 1
: 329      0461 1      ! Define bit patterns for calling OTS conversion routines.
: 330      0462 1
: 331      0463 1      K_IGN_BLKs = 1,
: 332      0464 1      K_ENB_UNDERFLOW = 4,
: 333      0465 1      K_IGN_TABS = 16,
: 334      0466 1      K_ENB_SCALE = 64,
: 335      0467 1
: 336      0468 1
: 337      0469 1      ! Bit map to use to set all arithmetic traps
: 338      0470 1
: 339      0471 1      K_SET_ARITHMETIC_TRAP = 32 + 64 + 128;
```

```

341 0472 1 ! Macros.
342 0473 1 !
343 0474 1 MACRO
344 0475 1
345 0476 1
346 0477 1 ! These macros define portions of intermediate data buffer.
347 0478 1 !
348 0479 1 LONG_1 = 0, 0, 32, 0 %,
349 0480 1 LONG_2 = 4, 0, 32, 0 %,
350 0481 1 LONG_3 = 8, 0, 32, 0 %,
351 0482 1 LONG_4 = 12, 0, 32, 0 %,
352 0483 1 LONG_5 = 16, 0, 32, 0 %,
353 0484 1 LONG_6 = 20, 0, 32, 0 %,
354 0485 1 LONG_7 = 24, 0, 32, 0 %,
355 0486 1 LONG_8 = 28, 0, 32, 0 %,
356 0487 1 S_LONG_1 = 0, 0, 32, 1 %,
357 0488 1 S_LONG_2 = 4, 0, 32, 1 %,
358 0489 1 S_BYTE_1 = 0, 0, 8, 1 %,
359 0490 1 BYTE_1 = 0, 0, 8, 0 %,
360 0491 1 BYTE_2 = 1, 0, 8, 0 %,
361 0492 1 S_WORD_1 = 0, 0, 16, 1 %,
362 0493 1 WORD_1 = 0, 0, 16, 0 %,
363 0494 1 WORD_2 = 2, 0, 16, 0 %,
364 0495 1 NIBBLE_1 = 0, 0, 4, 0 %,
365 0496 1
366 0497 1
367 0498 1 ! This macro calculates final states given the current state and the token.
368 0499 1 !
369 M 0500 1 FINAL_STATE (CLASS, DATA_TYPE) =
370 M 0501 1 (TK_MAX_DATA_TYPES *
371 M 0502 1 BEGIN
372 M 0503 1 CASE CLASS FROM K_MIN_CLASS TO K_MAX_CLASS OF
373 M 0504 1 SET
374 M 0505 1
375 M 0506 1 ! These are presently the only classes permitted.
376 M 0507 1 !
377 M 0508 1 [K_STATE1_CLASS_S]: 0;
378 M 0509 1 [K_STATE2_CLASS_D]: 1;
379 M 0510 1 [K_STATE4_CLASS_A]: 2;
380 M 0511 1 [K_STATE9_CLASS_SD]: 3;
381 M 0512 1 [K_STATE10_CLASS_NCA]: 4;
382 M 0513 1 [K_STATE11_CLASS_VS]: 5;
383 M 0514 1 [K_STATE13_CLASS_UBS]: 6;
384 M 0515 1 [INRANGE]:
385 M 0516 1 BEGIN
386 M 0517 1 $DBG_ERROR ('DBGCVTDX: invalid class');
387 M 0518 1 0
388 M 0519 1 END;
389 M 0520 1 TES
390 0521 1 END ) + DATA_TYPE) %,
391 0522 1
392 0523 1
393 0524 1 ! Again, the SRC and DST_INFO records are filled in by FIND_CVT_PATH
394 0525 1 ! so that information concerning the source and/or destination descriptors
395 0526 1 ! is readily available to DBG$CVT_DX_DX.
396 0527 1 !
397 0528 1 ! These macros are used for SRC_INFO or DST_INFO scale fields.

```

```

398 0529 1
399 0530 1 M_SCALE = 0, 0, 8, 1 %,
400 0531 1 M_BIN_SCALE = 7, 1, 1, 0 %,
401 0532 1
402 0533 1
403 0534 1 ! This macro is used for SRC_INFO or DST_INFO length field.
404 0535 1
405 0536 1 M_LEN = 5, 0, 16, 0 %,
406 0537 1
407 0538 1
408 0539 1 ! Define the start state.
409 0540 1
410 0541 1 START_STATE = VECTOR [K_MAX_CLASSES, BYTE, SIGNED] %,
411 0542 1
412 0543 1
413 0544 1 ! These MACROS are defined for the purpose of clarity, less typing, and anticipation
414 0545 1 ! of future support of BUILTINS.
415 0546 1
416 0547 1 ASHP = DBG$CVT_ASHP_R1 %,
417 0548 1 CMPH = DBG$CVT_CMPH_R1 %,
418 0549 1 CVTDH = DBG$CVT_CVTDR_R1 %,
419 0550 1 CVTHD = DBG$CVT_CVTHD_R1 %,
420 0551 1 CVTHF = DBG$CVT_CVTHF_R1 %,
421 0552 1 CVTHG = DBG$CVT_CVTHG_R1 %,
422 0553 1 CVTGH = DBG$CVT_CVTGH_R1 %,
423 0554 1 CVTLB = DBG$CVT_CVTLB_R1 %,
424 0555 1 CVTLH = DBG$CVT_CVTLH_R1 %,
425 0556 1 CVTLW = DBG$CVT_CVTLW_R1 %,
426 0557 1 CVTRDQ = DBG$CVT_CVTRDQ_R1 %,
427 0558 1 CVTRHL = DBG$CVT_CVTRHL_R1 %,
428 0559 1 CVTRHO = DBG$CVT_CVTRHO_R1 %,
429 0560 1 CVTRHQ = DBG$CVT_CVTRHQ_R1 %,
430 0561 1 CVTROUD = DBG$CVT_CVTROUD_R1 %,
431 0562 1 CVTROUH = DBG$CVT_CVTROUH_R1 %,
432 0563 1 DIVD2 = DBG$CVT_DIVD2_R1 %,
433 0564 1 DIVH2 = DBG$CVT_DIVH2_R1 %,
434 0565 1 DIVP = DBG$CVT_DIVP_R1 %,
435 0566 1 MULD2 = DBG$CVT_MULD2_R1 %,
436 0567 1 MULH2 = DBG$CVT_MULH2_R1 %,
437 0568 1 Mulp = DBG$CVT_MULP_R1 %,
438 0569 1
439 0570 1
440 0571 1 ! The following macros scale the intermediate data.
441 0572 1
442 0573 1 ! These macros scale the longword in INTMED_DATA buffer.
443 0574 1
444 M 0575 1 M_SCALE L_L =
445 M 0576 1 WHILE .BIN_SCALE GTR 0 DO
446 M 0577 1 BEGIN
447 M 0578 1 INTMED_DATA [LONG_1] = .INTMED_DATA [S_LONG_1]*2;
448 M 0579 1 BIN_SCALE = .BIN_SCALE - 1;
449 M 0580 1 END;
450 M 0581 1 WHILE .BIN_SCALE LSS 0 DO
451 M 0582 1 BEGIN
452 M 0583 1 INTMED_DATA [LONG_1] = .INTMED_DATA [S_LONG_1]/2;
453 M 0584 1 BIN_SCALE = .BIN_SCALE + 1;
454 M 0585 1 END;

```

```

455 M 0586 1 WHILE .SCALE GTR 0 DO
456 M 0587 1 BEGIN
457 M 0588 1 INTMED_DATA [LONG_1] = .INTMED_DATA [S_LONG_1]*10;
458 M 0589 1 SCALE = .SCALE - T;
459 M 0590 1 END;
460 M 0591 1 WHILE .SCALE LSS 0 DO
461 M 0592 1 BEGIN
462 M 0593 1 INTMED_DATA [LONG_1] = .INTMED_DATA [S_LONG_1]/10;
463 M 0594 1 SCALE = .SCALE + T;
464 M 0595 1 END
465 M 0596 1 X,
466 M 0597 1
467 M 0598 1
468 M 0599 1 ! Convert L to OU and scale it. INTMED_DATA is used for L and OU
469 M 0600 1 !
470 M 0601 1 M_SCALE_L_OU =
471 M 0602 1
472 M 0603 1 IF .INTMED_DATA [S_LONG_1] LSS 0
473 M 0604 1 THEN
474 M 0605 1 BEGIN
475 M 0606 1 INTMED_DATA [LONG_1] = ABS (.INTMED_DATA [S_LONG_1]);
476 M 0607 1 SRC_INFO [S_SIGN] = 1;
477 M 0608 1 END;
478 M 0609 1
479 M 0610 1 WHILE .SCALE GTR 0 DO
480 M 0611 1 BEGIN
481 M 0612 1 LIB$SCVT_SCALE_OU_UP_BY_10_R1 (INTMED_DATA);
482 M 0613 1 SCALE = .SCALE - T;
483 M 0614 1 END;
484 M 0615 1
485 M 0616 1 WHILE .SCALE LSS 0 DO
486 M 0617 1 BEGIN
487 M 0618 1 LIB$SCVT_SCALE_OU_DOWN_BY_1_R1 (INTMED_DATA);
488 M 0619 1 SCALE = .SCALE + T;
489 M 0620 1 END;
490 M 0621 1
491 M 0622 1 WHILE .BIN_SCALE GTR 0 DO
492 M 0623 1 BEGIN
493 M 0624 1 DBG$CVT_SCALE_OU_UP_BY_2_R1 (INTMED_DATA);
494 M 0625 1 BIN_SCALE = .BIN_SCALE - 1;
495 M 0626 1 END;
496 M 0627 1
497 M 0628 1 WHILE .BIN_SCALE LSS 0 DO
498 M 0629 1 BEGIN
499 M 0630 1 DBG$CVT_SCALE_OU_DOWN_BY_2_R1 (INTMED_DATA);
500 M 0631 1 BIN_SCALE = .BIN_SCALE + 1;
501 M 0632 1 END
502 M 0633 1 X,
503 M 0634 1
504 M 0635 1
505 M 0636 1
506 M 0637 1 ! Convert L to D, and scale it. INTMED_DATA buffer is used for L and D.
507 M 0638 1 !
508 M 0639 1 M_SCALE_L_D =
509 M 0640 1 CVTCD (INTMED_DATA, INTMED_DATA);
510 M 0641 1
511 M 0642 1 WHILE .BIN_SCALE GTR 0 DO
```



```
512 M 0643 1 BEGIN
513 M 0644 1 MUL2 (UPLIT (%D'2'), INTMED_DATA);
514 M 0645 1 BIN_SCALE = .BIN_SCALE - 1;
515 M 0646 1 END;
516 M 0647 1
517 M 0648 1 WHILE .BIN_SCALE LSS 0 DO
518 M 0649 1 BEGIN
519 M 0650 1 DIV2 (UPLIT (%D'2'), INTMED_DATA);
520 M 0651 1 BIN_SCALE = .BIN_SCALE + 1;
521 M 0652 1 END;
522 M 0653 1
523 M 0654 1 WHILE .SCALE GTR 0 DO
524 M 0655 1 BEGIN
525 M 0656 1 MUL2 (UPLIT (%D'10'), INTMED_DATA);
526 M 0657 1 SCALE = .SCALE - 1;
527 M 0658 1 END;
528 M 0659 1
529 M 0660 1 WHILE .SCALE LSS 0 DO
530 M 0661 1 BEGIN
531 M 0662 1 DIV2 (UPLIT (%D'10'), INTMED_DATA);
532 M 0663 1 SCALE = .SCALE + 1;
533 M 0664 1 END
534 M 0665 1
535 M 0666 1 %,
536 M 0667 1
537 M 0668 1
538 M 0669 1 ! Convert L to P, and scale it. INTMED_DATA is the buffer for L and P.
539 M 0670 1 !
540 M 0671 1 M_SCALE_L_P =
541 M 0672 1 IF .INTMED_DATA [S_LONG_1] LSS 0 THEN SRC_INFO [S_SIGN] = 1;
542 M 0673 1
543 M 0674 1 NO_DIGITS = 31;
544 M 0675 1 CVTLP (INTMED_DATA, NO_DIGITS, INTMED_DATA);
545 M 0676 1
546 M 0677 1 IF .SCALE NEQ 0
547 M 0678 1 THEN
548 M 0679 1 BEGIN
549 M 0680 1 MOV (NO_DIGITS, INTMED_DATA, TEMP_BUF1);
550 M 0681 1 IF .CVT_ROUND_FLAG
551 M 0682 1 THEN
552 M 0683 1 ASHP (SCALE, NO_DIGITS, TEMP_BUF1, %REF (5), NO_DIGITS, INTMED_DATA)
553 M 0684 1 ELSE
554 M 0685 1 ASHP (SCALE, NO_DIGITS, TEMP_BUF1, %REF (0), NO_DIGITS, INTMED_DATA);
555 M 0686 1
556 M 0687 1 END;
557 M 0688 1
558 M 0689 1 WHILE .BIN_SCALE GTR 0 DO
559 M 0690 1 BEGIN
560 M 0691 1 MOV (NO_DIGITS, INTMED_DATA, TEMP_BUF1);
561 M 0692 1 MULP (%REF (1), UPLIT (%P'2'), NO_DIGITS, TEMP_BUF1, NO_DIGITS, INTMED_DATA);
562 M 0693 1 BIN_SCALE = .BIN_SCALE - 1;
563 M 0694 1 END;
564 M 0695 1
565 M 0696 1 WHILE .BIN_SCALE LSS 0 DO
566 M 0697 1 BEGIN
567 M 0698 1 MOV (NO_DIGITS, INTMED_DATA, TEMP_BUF1);
568 M 0699 1
```

```

569      DIVP (%REF (1), UPLIT (%P'2'), NO_DIGITS, TEMP_BUF1, NO_DIGITS, INTMED_DATA);
570      BIN_SCALE = .BIN_SCALE + 1;
571      END
572      M 0703 1
573      0704 1
574      0705 1
575      0706 1
576      0707 1
577      0708 1
578      M 0709 1
579      M 0710 1
580      M 0711 1
581      M 0712 1
582      M 0713 1
583      M 0714 1
584      M 0715 1
585      M 0716 1
586      M 0717 1
587      M 0718 1
588      M 0719 1
589      M 0720 1
590      M 0721 1
591      M 0722 1
592      M 0723 1
593      M 0724 1
594      M 0725 1
595      M 0726 1
596      M 0727 1
597      M 0728 1
598      M 0729 1
599      M 0730 1
600      M 0731 1
601      M 0732 1
602      M 0733 1
603      M 0734 1
604      0735 1
605      0736 1
606      0737 1
607      0738 1
608      0739 1
609      M 0740 1
610      M 0741 1
611      M 0742 1
612      M 0743 1
613      M 0744 1
614      M 0745 1
615      M 0746 1
616      M 0747 1
617      M 0748 1
618      M 0749 1
619      M 0750 1
620      M 0751 1
621      M 0752 1
622      M 0753 1
623      M 0754 1
624      M 0755 1
625      M 0756 1

      ! Scale the OU in INTMED_DATA buffer.
      M_SCALE_OU_OU =
      WHILE .SCALE GTR 0 DO
      BEGIN
      LIB$SCVT_SCALE_OU_UP_BY_10_R1 (INTMED_DATA);
      SCALE = .SCALE - 1;
      END;

      WHILE .SCALE LSS 0 DO
      BEGIN
      LIB$SCVT_SCALE_OU_DOWN_BY_10_R1 (INTMED_DATA);
      SCALE = .SCALE + 1;
      END;

      WHILE .BIN_SCALE GTR 0 DO
      BEGIN
      DBG$CVT_SCALE_OU_UP_BY_2_R1 (INTMED_DATA);
      BIN_SCALE = .BIN_SCALE - 1;
      END;

      WHILE .BIN_SCALE LSS 0 DO
      BEGIN
      DBG$CVT_SCALE_OU_DOWN_BY_2_R1 (INTMED_DATA);
      BIN_SCALE = .BIN_SCALE + 1;
      END;

      ! Convert OU to D, and scale it. INTMED_DATA is used for OU and D.
      M_SCALE_OU_D =
      CVTROUD (INTMED_DATA, TEMP_BUF1);
      CHSMOVE (8, TEMP_BUF1, INTMED_DATA);

      WHILE .BIN_SCALE GTR 0 DO
      BEGIN
      MUL2 (UPLIT (%D'2'), INTMED_DATA);
      BIN_SCALE = .BIN_SCALE - 1;
      END;

      WHILE .BIN_SCALE LSS 0 DO
      BEGIN
      DIV2 (UPLIT (%D'2'), INTMED_DATA);
      BIN_SCALE = .BIN_SCALE + 1;
      END;

      WHILE .SCALE GTR 0 DO
```

```

626      M 0757 1      BEGIN
627      M 0758 1      MULH2 (UPLIT (%D'10'), INTMED_DATA);
628      M 0759 1      SCALE = .SCALE - 1;
629      M 0760 1      END;
630      M 0761 1
631      M 0762 1      WHILE .SCALE LSS 0 DO
632      M 0763 1      BEGIN
633      M 0764 1      DIVH2 (UPLIT (%D'10'), INTMED_DATA);
634      M 0765 1      SCALE = .SCALE + 1;
635      M 0766 1      END
636      M 0767 1
637      M 0768 1      %,
638      M 0769 1
639      M 0770 1      ! Convert OU to H, and scale it. INTMED_DATA is used for OU and H.
640      M 0771 1      !
641      M 0772 1      M_SCALE OU H =
642      M 0773 1      CVTROUR (INTMED_DATA, TEMP_BUF1);
643      M 0774 1      CH$MOVE (16, TEMP_BUF1, INTMED_DATA);
644      M 0775 1
645      M 0776 1      WHILE .BIN_SCALE GTR 0 DO
646      M 0777 1      BEGIN
647      M 0778 1      MULH2 (UPLIT (%H'2'), INTMED_DATA);
648      M 0779 1      BIN_SCALE = .BIN_SCALE - 1;
649      M 0780 1      END;
650      M 0781 1
651      M 0782 1      WHILE .BIN_SCALE LSS 0 DO
652      M 0783 1      BEGIN
653      M 0784 1      DIVH2 (UPLIT (%H'2'), INTMED_DATA);
654      M 0785 1      BIN_SCALE = .BIN_SCALE + 1;
655      M 0786 1      END;
656      M 0787 1
657      M 0788 1      WHILE .SCALE GTR 0 DO
658      M 0789 1      BEGIN
659      M 0790 1      MULH2 (UPLIT (%H'10'), INTMED_DATA);
660      M 0791 1      SCALE = .SCALE - 1;
661      M 0792 1      END;
662      M 0793 1
663      M 0794 1      WHILE .SCALE LSS 0 DO
664      M 0795 1      BEGIN
665      M 0796 1      DIVH2 (UPLIT (%H'10'), INTMED_DATA);
666      M 0797 1      SCALE = .SCALE + 1;
667      M 0798 1      END
668      M 0799 1
669      M 0800 1      %,
670      M 0801 1
671      M 0802 1      ! Convert L to H, and scale it. INTMED_DATA is used for L and H.
672      M 0803 1      !
673      M 0804 1      M_SCALE L H =
674      M 0805 1      CVTCH- (INTMED_DATA, INTMED_DATA);
675      M 0806 1
676      M 0807 1      WHILE .BIN_SCALE GTR 0 DO
677      M 0808 1      BEGIN
678      M 0809 1      MULH2 (UPLIT (%H'2'), INTMED_DATA);
679      M 0810 1      BIN_SCALE = .BIN_SCALE - 1;
680      M 0811 1      END;
681      M 0812 1
682      M 0813 1
```

```

683      M 0814 1
684      M 0815 1
685      M 0816 1
686      M 0817 1
687      M 0818 1
688      M 0819 1
689      M 0820 1
690      M 0821 1
691      M 0822 1
692      M 0823 1
693      M 0824 1
694      M 0825 1
695      M 0826 1
696      M 0827 1
697      M 0828 1
698      M 0829 1
699      M 0830 1
700      M 0831 1
701      M 0832 1
702      M 0833 1
703      M 0834 1
704      M 0835 1
705      M 0836 1
706      M 0837 1
707      M 0838 1
708      M 0839 1
709      M 0840 1
710      M 0841 1
711      M 0842 1
712      M 0843 1
713      M 0844 1
714      M 0845 1
715      M 0846 1
716      M 0847 1
717      M 0848 1
718      M 0849 1
719      M 0850 1
720      M 0851 1
721      M 0852 1
722      M 0853 1
723      M 0854 1
724      M 0855 1
725      M 0856 1
726      M 0857 1
727      M 0858 1
728      M 0859 1
729      M 0860 1
730      M 0861 1
731      M 0862 1
732      M 0863 1
733      M 0864 1
734      M 0865 1
735      M 0866 1
736      M 0867 1
737      M 0868 1
738      M 0869 1
739      M 0870 1

      WHILE .BIN_SCALE LSS 0 DO
      BEGIN
      DIVH2 (UPLIT (%H'2'), INTMED_DATA);
      BIN_SCALE = .BIN_SCALE + 1;
      END;

      WHILE .SCALE GTR 0 DO
      BEGIN
      MULH2 (UPLIT (%H'10'), INTMED_DATA);
      SCALE = .SCALE - 1;
      END;

      WHILE .SCALE LSS 0 DO
      BEGIN
      DIVH2 (UPLIT (%H'10'), INTMED_DATA);
      SCALE = .SCALE + 1;
      END

      %,

      ! Scale D in INTMED_DATA.
      !
      M_SCALE_D_D =

      WHILE .BIN_SCALE GTR 0 DO
      BEGIN
      MULD2 (UPLIT (%D'2'), INTMED_DATA);
      MULD2 (UPLIT (%D'2'), INTMED_DATA+8);
      BIN_SCALE = .BIN_SCALE - 1;
      END;

      WHILE .BIN_SCALE LSS 0 DO
      BEGIN
      DIVD2 (UPLIT (%D'2'), INTMED_DATA);
      DIVD2 (UPLIT (%D'2'), INTMED_DATA+8);
      BIN_SCALE = .BIN_SCALE + 1;
      END;

      WHILE .SCALE GTR 0 DO
      BEGIN
      MULD2 (UPLIT (%D'10'), INTMED_DATA);
      MULD2 (UPLIT (%D'10'), INTMED_DATA+8);
      SCALE = .SCALE - 1;
      END;

      WHILE .SCALE LSS 0 DO
      BEGIN
      DIVD2 (UPLIT (%D'10'), INTMED_DATA);
      DIVD2 (UPLIT (%D'10'), INTMED_DATA+8);
      SCALE = .SCALE + 1;
      END

      %,

```

```

740      0871 1      ! Convert D to H, and scale it. INTMED_DATA is used for D and H.
741      0872 1
742      M 0873 1      M_SCALE_D_H =
743      M 0874 1      CVDH (INTMED_DATA, TEMP_BUF1);
744      M 0875 1      CVDH (INTMED_DATA+8, INTMED_DATA+16);
745      M 0876 1      CH$MOVE (16, TEMP_BUF1, INTMED_DATA);
746      M 0877 1
747      M 0878 1      WHILE .BIN_SCALE GTR 0 DO
748      M 0879 1      BEGIN
749      M 0880 1      MULH2 (UPLIT (%H'2'), INTMED_DATA);
750      M 0881 1      MULH2 (UPLIT (%H'2'), INTMED_DATA+16);
751      M 0882 1      BIN_SCALE = .BIN_SCALE - 1;
752      M 0883 1      END;
753      M 0884 1
754      M 0885 1      WHILE .BIN_SCALE LSS 0 DO
755      M 0886 1      BEGIN
756      M 0887 1      DIVH2 (UPLIT (%H'2'), INTMED_DATA);
757      M 0888 1      DIVH2 (UPLIT (%H'2'), INTMED_DATA+16);
758      M 0889 1      BIN_SCALE = .BIN_SCALE + 1;
759      M 0890 1      END;
760      M 0891 1
761      M 0892 1      WHILE .SCALE GTR 0 DO
762      M 0893 1      BEGIN
763      M 0894 1      MULH2 (UPLIT (%H'10'), INTMED_DATA);
764      M 0895 1      MULH2 (UPLIT (%H'10'), INTMED_DATA+16);
765      M 0896 1      SCALE = .SCALE - 1;
766      M 0897 1      END;
767      M 0898 1
768      M 0899 1      WHILE .SCALE LSS 0 DO
769      M 0900 1      BEGIN
770      M 0901 1      DIVH2 (UPLIT (%H'10'), INTMED_DATA);
771      M 0902 1      DIVH2 (UPLIT (%H'10'), INTMED_DATA+16);
772      M 0903 1      SCALE = .SCALE + 1;
773      M 0904 1      END;
774      M 0905 1
775      0906 1      X.
776      0907 1
777      0908 1
778      0909 1      ! Convert G to H, and scale it. INTMED_DATA is used for G and H.
779      0910 1
780      M 0911 1      M_SCALE_G_H =
781      M 0912 1      BEGIN
782      M 0913 1      CVTGH (INTMED_DATA, TEMP_BUF1);
783      M 0914 1      CVTGH (INTMED_DATA+8, INTMED_DATA+16);
784      M 0915 1      CH$MOVE (16, TEMP_BUF1, INTMED_DATA);
785      M 0916 1
786      M 0917 1      WHILE .BIN_SCALE GTR 0 DO
787      M 0918 1      BEGIN
788      M 0919 1      MULH2 (UPLIT (%H'2'), INTMED_DATA);
789      M 0920 1      MULH2 (UPLIT (%H'2'), INTMED_DATA+16);
790      M 0921 1      BIN_SCALE = .BIN_SCALE - 1;
791      M 0922 1      END;
792      M 0923 1
793      M 0924 1      WHILE .BIN_SCALE LSS 0 DO
794      M 0925 1      BEGIN
795      M 0926 1      DIVH2 (UPLIT (%H'2'), INTMED_DATA);
796      M 0927 1      DIVH2 (UPLIT (%H'2'), INTMED_DATA+16);

```

```

797 M 0928 1      BIN_SCALE = .BIN_SCALE + 1;
798 M 0929 1      END;
799 M 0930 1
800 M 0931 1      WHILE .SCALE GTR 0 DO
801 M 0932 1      BEGIN
802 M 0933 1      MULH2 (UPLIT (%H'10'), INTMED_DATA);
803 M 0934 1      MULH2 (UPLIT (%H'10'), INTMED_DATA+16);
804 M 0935 1      SCALE = .SCALE - 1;
805 M 0936 1      END;
806 M 0937 1
807 M 0938 1      WHILE .SCALE LSS 0 DO
808 M 0939 1      BEGIN
809 M 0940 1      DIVH2 (UPLIT (%H'10'), INTMED_DATA);
810 M 0941 1      DIVH2 (UPLIT (%H'10'), INTMED_DATA+16);
811 M 0942 1      SCALE = .SCALE + 1;
812 M 0943 1      END;
813 M 0944 1      END
814 M 0945 1
815 M 0946 1      %,
816 M 0947 1
817 M 0948 1      ! Scale H in INTMED_DATA.
818 M 0949 1      !
819 M 0950 1      M_SCALE_H_H =
820 M 0951 1
821 M 0952 1      WHILE .BIN_SCALE GTR 0 DO
822 M 0953 1      BEGIN
823 M 0954 1      MULH2 (UPLIT (%H'2'), INTMED_DATA);
824 M 0955 1      MULH2 (UPLIT (%H'2'), INTMED_DATA+16);
825 M 0956 1      BIN_SCALE = .BIN_SCALE - 1;
826 M 0957 1      END;
827 M 0958 1
828 M 0959 1      WHILE .BIN_SCALE LSS 0 DO
829 M 0960 1      BEGIN
830 M 0961 1      DIVH2 (UPLIT (%H'2'), INTMED_DATA);
831 M 0962 1      DIVH2 (UPLIT (%H'2'), INTMED_DATA+16);
832 M 0963 1      BIN_SCALE = .BIN_SCALE + 1;
833 M 0964 1      END;
834 M 0965 1
835 M 0966 1      WHILE .SCALE GTR 0 DO
836 M 0967 1      BEGIN
837 M 0968 1      MULH2 (UPLIT (%H'10'), INTMED_DATA);
838 M 0969 1      MULH2 (UPLIT (%H'10'), INTMED_DATA+16);
839 M 0970 1      SCALE = .SCALE - 1;
840 M 0971 1      END;
841 M 0972 1
842 M 0973 1      WHILE .SCALE LSS 0 DO
843 M 0974 1      BEGIN
844 M 0975 1      DIVH2 (UPLIT (%H'10'), INTMED_DATA);
845 M 0976 1      DIVH2 (UPLIT (%H'10'), INTMED_DATA+16);
846 M 0977 1      SCALE = .SCALE + 1;
847 M 0978 1      END
848 M 0979 1
849 M 0980 1
850 M 0981 1      %,
851 M 0982 1
852 M 0983 1
853 M 0984 1      ! Scale P in INTMED_DATA

```

```
854 0985 1
855 M 0986 1
856 M 0987 1
857 M 0988 1
858 M 0989 1
859 M 0990 1
860 M 0991 1
861 M 0992 1
862 M 0993 1
863 M 0994 1
864 M 0995 1
865 M 0996 1
866 M 0997 1
867 M 0998 1
868 M 0999 1
869 M 1000 1
870 M 1001 1
871 M 1002 1
872 M 1003 1
873 M 1004 1
874 M 1005 1
875 M 1006 1
876 M 1007 1
877 M 1008 1
878 M 1009 1
879 M 1010 1
880 M 1011 1
881 M 1012 1
882 M 1013 1
883 M 1014 1
884 M 1015 1
885 M 1016 1
886 M 1017 1
887 M 1018 1
888 M 1019 1
889 M 1020 1
890 M 1021 1
891 M 1022 1
892 M 1023 1
893 M 1024 1
894 M 1025 1
895 M 1026 1
896 M 1027 1
897 M 1028 1
898 M 1029 1
899 M 1030 1
900 M 1031 1
901 M 1032 1
902 M 1033 1
903 M 1034 1
904 M 1035 1
905 M 1036 1
906 M 1037 1
907 M 1038 1
908 M 1039 1
909 M 1040 1
910 M 1041 1
```

```
!
M_SCALE_P_P =
NO_DIGITS = .SRC_INFO [S_LEN];
IF (CMPP (NO_DIGITS, INTMED_DATA, %REF (1), .PACK_ZERO) LSS 0) THEN SRC_INFO [S_SIGN] = 1;
IF .SCALE NEQ 0
THEN
BEGIN
MOVP (NO_DIGITS, INTMED_DATA, TEMP_BUF1);
IF .CVT_ROUND_FLAG
THEN
ASHP (SCALE, NO_DIGITS, TEMP_BUF1, %REF (5), NO_DIGITS, INTMED_DATA)
ELSE
ASHP (SCALE, NO_DIGITS, TEMP_BUF1, %REF (0), NO_DIGITS, INTMED_DATA);
END;
```

I (PS) added the following code, because, if I deposit a packed decimal number 999.888 into a 4 digits decimal number scaled -2, I want to get a result of 99.88, instead of later on I will get overflow error, and have nothing as result. Check to see if the significant digits of source is greater than the significant digits of the destination.

This piece of code is used only if both operands are packed.

```
IF (.SOURCE[DSC$W_LENGTH] + .SOURCE[DSC$B_SCALE] GTR
.DESTINATION[DSC$W_LENGTH] + .DESTINATION[DSC$B_SCALE]) AND
(.SOURCE[DSC$B_DTYPE] EQL DSC$K_DTYPE_P AND
.DESTINATION[DSC$B_DTYPE] EQL DSC$K_DTYPE_P)
```

```
THEN
BEGIN
LOCAL
HIGH_NIBBLE_PTR: REF VECTOR[BYTE],
LOW_NIBBLE_PTR: REF VECTOR[BYTE];
```

! Point to the last digits.

```
HIGH_NIBBLE_PTR = INTMED_DATA + 16 - 1;
```

! Backup the pointer to the significant digit needs to be truncated. Zero out everything before that.

```
LOW_NIBBLE_PTR = .HIGH_NIBBLE_PTR -
(.DESTINATION[DSC$W_LENGTH] / 2 + 1) + 1;
```

! If destination digits is even, we need to zero out one nibble. Note: this may be already zero.

```
IF (.DESTINATION[DSC$W_LENGTH] MOD 2) EQL 0
THEN
LOW_NIBBLE_PTR[0] = .LOW_NIBBLE_PTR[0] AND %X'0F';
```

```

911 M 1042 1
912 M 1043 1
913 M 1044 1
914 M 1045 1
915 M 1046 1
916 M 1047 1
917 M 1048 1
918 M 1049 1
919 M 1050 1
920 M 1051 1
921 M 1052 1
922 M 1053 1
923 M 1054 1
924 M 1055 1
925 M 1056 1
926 M 1057 1
927 M 1058 1
928 M 1059 1
929 M 1060 1
930 M 1061 1
931 M 1062 1
932 M 1063 1
933 M 1064 1
934 M 1065 1
935 M 1066 1
936 M 1067 1
937 M 1068 1
938 M 1069 1
939 M 1070 1
940 M 1071 1
941 M 1072 1
942 M 1073 1
943 M 1074 1
944 M 1075 1
945 M 1076 1
946 M 1077 1
947 M 1078 1
948 M 1079 1
949 M 1080 1
950 M 1081 1
951 M 1082 1
952 M 1083 1
953 M 1084 1
954 M 1085 1
955 M 1086 1
956 M 1087 1
957 M 1088 1
958 M 1089 1
959 M 1090 1
960 M 1091 1
961 M 1092 1
962 M 1093 1
963 M 1094 1
964 M 1095 1
965 M 1096 1
966 M 1097 1
967 M 1098 1

! Zero out everything before it. Note: this may be already
! zero.
LOW_NIBBLE_PTR = .LOW_NIBBLE_PTR - 1;
WHILE .LOW_NIBBLE_PTR GEQ INTMED_DATA DO
  BEGIN
    LOW_NIBBLE_PTR[0] = %X'00';
    LOW_NIBBLE_PTR = .LOW_NIBBLE_PTR - 1;
  END;

SIGNAL(DBG$_INUMTRUNC, 1, .DBG$GL_OPCODE_NAME);
END

%,

! Convert P to OU, and scale it. INTMED_DATA is used for P and OU.
M_SCALE P OU =
  NO_DIGITS = .SRC_INFO [S_LEN];
  CVT$P (NO_DIGITS, INTMED_DATA, NO_DIGITS, TEMP_BUF1);
  CLASS_S_DESC [DSC$W_LENGTH] = .NO_DIGITS + 1;
  CLASS_S_DESC [DSC$A_POINTER] = TEMP_BUF1;
  OT$SCVT_T_H (CLASS_S_DESC, TEMP_BUF2);

  IF .TEMP_BUF2 [0, 15, 1, 0]
  THEN
    BEGIN
      TEMP_BUF2 [0, 15, 1, 0] = 0;
      SRC_INFO [S_SIGN] = 1;
    END;

  CVTRHO (TEMP_BUF2, INTMED_DATA);

  WHILE .SCALE GTR 0 DO
    BEGIN
      LIB$SCVT_SCALE_OU_UP_BY_10_R1 (INTMED_DATA);
      SCALE = .SCALE - 1;
    END;

  WHILE .SCALE LSS 0 DO
    BEGIN
      LIB$SCVT_SCALE_OU_DOWN_BY_10_R1 (INTMED_DATA);
      SCALE = .SCALE + 1;
    END;

  WHILE .BIN_SCALE GTR 0 DO
    BEGIN
      DBG$CVT_SCALE_OU_UP_BY_2_R1 (INTMED_DATA);
      BIN_SCALE = .BIN_SCALE - 1;
    END;

  WHILE .BIN_SCALE LSS 0 DO
    BEGIN
      DBG$CVT_SCALE_OU_DOWN_BY_2_R1 (INTMED_DATA);
```



```

: 968      M 1099 1      BIN_SCALE = .BIN_SCALE + 1;
: 969      M 1100 1      END
: 970      M 1101 1
: 971      M 1102 1      %.
: 972      M 1103 1
: 973      M 1104 1
: 974      M 1105 1      ! Convert P to D, and scale it. INTMED_DATA is used for P and D.
: 975      M 1106 1
: 976      M 1107 1      M_SCALE_P_D =
: 977      M 1108 1      NO_DIGITS = .SRC_INFO [S_LEN];
: 978      M 1109 1
: 979      M 1110 1      ! In the case of scaled packed, we need to get the scale this way.
: 980      M 1111 1
: 981      M 1112 1      IF .SOURCE[DSC$B_CLASS] EQL DSC$K_CLASS_SD
: 982      M 1113 1      THEN
: 983      M 1114 1          SCALE = - .SOURCE[DSC$B_SCALE];
: 984      M 1115 1
: 985      M 1116 1      CVTIPS (NO DIGITS, INTMED_DATA, NO_DIGITS, TEMP_BUF1);
: 986      M 1117 1      CLASS_S_DESC [DSC$W_LENGTH] = .NO_DIGITS + 1;
: 987      M 1118 1      CLASS_S_DESC [DSC$A_POINTER] = TEMP_BUF1;
: 988      M 1119 1      STATUS = OTSSCVT_T_D (CLASS_S_DESC, INTMED_DATA, 0, .SCALE, (K_ENB_UNDERFLOW OR K_ENB_SCALE));
: 989      M 1120 1
: 990      M 1121 1      IF NOT .STATUS
: 991      M 1122 1      THEN
: 992      M 1123 1          IF .SCALE LSS 0
: 993      M 1124 1              THEN SIGNAL (DBG$IFLTUND, 1, .DBG$GL_OPCODE_NAME)
: 994      M 1125 1              ELSE SIGNAL (DBG$_FLTOVF, 1, .DBG$GL_OPCODE_NAME); %.
: 995      M 1126 1
: 996      M 1127 1
: 997      M 1128 1      ! Convert P to H, and scale it. INTMED_DATA is used for P and H.
: 998      M 1129 1
: 999      M 1130 1      M_SCALE_P_H =
1000      M 1131 1      NO_DIGITS = .SRC_INFO [S_LEN];
1001      M 1132 1
1002      M 1133 1      ! In the case of scaled packed, we need to get the scale this way.
1003      M 1134 1
1004      M 1135 1      IF .SOURCE[DSC$B_CLASS] EQL DSC$K_CLASS_SD
1005      M 1136 1      THEN
1006      M 1137 1          SCALE = - .SOURCE[DSC$B_SCALE];
1007      M 1138 1
1008      M 1139 1      CVTIPS (NO DIGITS, INTMED_DATA, NO_DIGITS, TEMP_BUF1);
1009      M 1140 1      CLASS_S_DESC [DSC$W_LENGTH] = .NO_DIGITS + 1;
1010      M 1141 1      CLASS_S_DESC [DSC$A_POINTER] = TEMP_BUF1;
1011      M 1142 1      STATUS = OTSSCVT_T_H (CLASS_S_DESC, INTMED_DATA, 0, .SCALE, (K_ENB_UNDERFLOW OR K_ENB_SCALE));
1012      M 1143 1
1013      M 1144 1      IF NOT .STATUS
1014      M 1145 1      THEN
1015      M 1146 1          IF .SCALE LSS 0
1016      M 1147 1              THEN SIGNAL (DBG$IFLTUND, 1, .DBG$GL_OPCODE_NAME)
1017      M 1148 1              ELSE SIGNAL (DBG$_FLTOVF, 1, .DBG$GL_OPCODE_NAME); %.

```

```

1019 1149 1 ! Structure and Field Definitions.
1020 1150 1
1021 1151 1 STATES is a structure into which go all the states other than the first.
1022 1152 1 The final states and the states that never get used (such as the states that
1023 1153 1 contain non-supported CLASSES) will not be in this structure.
1024 1154 1
1025 1155 1 STRUCTURE
1026 1156 1 STATES [STATE, TOKEN] =
1027 1157 1 [K_ACTUAL_CLASSES*K_MAX_DATA_TYPES]
1028 1158 3 (STATES + (K_MAX_DATA_TYPES*
1029 1159 4 BEGIN
1030 1160 4 CASE STATE FROM K_MIN_CLASS TO K_MAX_CLASS OF
1031 1161 4 SET
1032 1162 4 [K_STATE1_CLASS_S]: 0;
1033 1163 4 [K_STATE2_CLASS_D]: 1;
1034 1164 4 [K_STATE4_CLASS_A]: 2;
1035 1165 4 [K_STATE9_CLASS_SD]: 3;
1036 1166 4 [K_STATE10_CLASS_NCA]: 4;
1037 1167 4 [K_STATE11_CLASS_VS]: 5;
1038 1168 4 [K_STATE13_CLASS_UBS]: 6;
1039 1169 4 [INRANGE, OUTRANGE]:
1040 1170 5 BEGIN
1041 1171 5 $DBG_ERROR ('DBGCVTDX: invalid class');
1042 1172 5 0
1043 1173 4 END;
1044 1174 4 TES
1045 1175 4 END
1046 1176 1 ) + TOKEN)<0, %BPUNIT, 1>;
1047 1177 1
1048 1178 1
1049 1179 1 ! SRC and DST INFO record fields.
1050 1180 1
1051 1181 1 FIELD
1052 1182 1 SRC_INFO_FIELDS =
1053 1183 1 SET
1054 1184 1 S_SCALE = [0, 0, 8, 1],
1055 1185 1 S_POINTER = [1, 0, 32, 0],
1056 1186 1 S_LEN = [5, 0, 16, 0],
1057 1187 1 S_SIGN = [7, 0, 1, 0],
1058 1188 1 S_BIN_SCALE = [7, 1, 1, 0] ! Flag indicating scale is binary
1059 1189 1 TES;
1060 1190 1
1061 1191 1 FIELD
1062 1192 1 DST_INFO_FIELDS =
1063 1193 1 SET
1064 1194 1 D_SCALE = [0, 0, 8, 1],
1065 1195 1 D_LEN = [5, 0, 16, 0],
1066 1196 1 D_BIN_SCALE = [7, 1, 1, 0] ! Flag indicating scale is binary
1067 1197 1 TES;

```

```
1069 1198 1 | State Table.
1070 1199 1 |
1071 1200 1 | Start States (all classes). CLASS_TABLE.
1072 1201 1 | These are the start state entries.
1073 1202 1 | For each CLASS in the standard there is an entry here. They are:
1074 1203 1 |       Z      S      D      V      A
1075 1204 1 |       .P      none .J      none .SD
1076 1205 1 |       .NCA     .VS     .VSA     .UBS     .UBA.
1077 1206 1 |
1078 1207 1 | BIND
1079 1208 1 | CLASS_TABLE = UPLIT BYTE
1080 1209 1 | % ( State zero. All classes. )%
1081 1210 1 | (K_UNSCALAROU,DSC$K_CLASS_S,DSC$K_CLASS_D,K_UNSCALAROU,DSC$K_CLASS_A
1082 1211 1 | ,K_UNSCALAROU,K_UNSCLASTA,K_UNSCALAROU,K_UNSCLASTA,DSC$K_CLASS_SD
1083 1212 1 | ,DSC$K_CLASS_NCA,DSC$K_CLASS_VS,K_UNSCALAROU,DSC$K_CLASS_UBS,R_UNSCALAROU): START_STATE;
1084 1213 1 |
1085 1214 1 |
1086 1215 1 | Remaining States. DTYPE_TABLE.
1087 1216 1 |
1088 1217 1 | This is the rest of the state table. It is separate for space efficiency.
1089 1218 1 | Each state contains entries for each data type supported by the standard.
1090 1219 1 | Note that for space efficiency the final states are not in the vector.
1091 1220 1 | Also since each state represents a supported CLASS, if a CLASS is not
1092 1221 1 | supported (by the standard or routine), then the state has no entry in
1093 1222 1 | the vector. The index table for the vector will index to the proper place
1094 1223 1 | in the vector below.
1095 1224 1 | The table below shows graphically what descriptors are valid.
1096 1225 1 |
1097 1226 1 |
1098 1227 1 |           DSC$K_DTYPE_x
1099 1228 1 |           BU WU LU B W L Q F D G H T NO NL NLO NR NRO NZ P VT AC AZ TF V SV VU SVU
1100 1229 1 | DSC$K_CLASS_S      x x x x x x x x x x x x x x x x x x x x x x x x
1101 1230 1 | DSC$K_CLASS_D      x
1102 1231 1 | DSC$K_CLASS_SD      x x x x x x x x x x x x x x x x x x
1103 1232 1 | DSC$K_CLASS_VS      x
1104 1233 1 | DSC$K_CLASS_A      x
1105 1234 1 | DSC$K_CLASS_NCA     x
1106 1235 1 | DSC$K_CLASS_UBS      x
1107 1236 1 |
1108 1237 1 | Note that these data types are hard coded in (zero based vector, and position
1109 1238 1 | of each data type is determined by the value of the symbol), so if data type
1110 1239 1 | values are ever rearranged this table must be rearranged.
1111 1240 1 |
1112 1241 1 | BIND
1113 1242 1 | DTYPE_TABLE = UPLIT BYTE
1114 1243 1 | % ( State zero. Class z. )%
1115 1244 1 | % ( State one. Class s. )%
1116 1245 1 | (K_UNSDTYROU,DSC$K_DTYPE_V,DSC$K_DTYPE_BU,DSC$K_DTYPE_WU,DSC$K_DTYPE_LU
1117 1246 1 | ,DSC$K_DTYPE_QU,DSC$K_DTYPE_B,DSC$K_DTYPE_W,DSC$K_DTYPE_L,DSC$K_DTYPE_Q
1118 1247 1 | ,DSC$K_DTYPE_F,DSC$K_DTYPE_D,DSC$K_DTYPE_FC,DSC$K_DTYPE_DC,DSC$K_DTYPE_T
1119 1248 1 | ,DSC$K_DTYPE_NU,DSC$K_DTYPE_NL,DSC$K_DTYPE_NLO,DSC$K_DTYPE_NR,DSC$K_DTYPE_NRO
1120 1249 1 | ,DSC$K_DTYPE_NZ,DSC$K_DTYPE_P,DSC$K_DTYPE_ZI,K_UNSDTYROU,K_UNSDTYROU
1121 1250 1 | ,K_UNSDTYROU,DSC$K_DTYPE_O,DSC$K_DTYPE_G,DSC$K_DTYPE_H,DSC$K_DTYPE_GC
1122 1251 1 | ,DSC$K_DTYPE_HC,K_UNSDTYROU,K_UNSDTYROU,K_UNSDTYROU,K_UNSDTYROU
1123 1252 1 | ,K_UNSDTYROU,K_UNSDTYROU,K_UNSDTYROU,K_UNSDTYROU,K_UNSDTYROU,DSC$K_DTYPE_TF
1124 1253 1 | ,DSC$K_DTYPE_SF,K_UNSDTYROU
1125 1254 1 | % ( State two. Class d. )%
```

```

: 1126      1255 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1127      1256 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1128      1257 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,DSC$K_DTYPE_T
: 1129      1258 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1130      1259 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1131      1260 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1132      1261 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1133      1262 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1134      1263 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1135      1264 1      %( State three. (Class v. )%
: 1136      1265 1      %( State four. (Class a. )%
: 1137      1266 1      ,K_UNSDTYROU,K_UNSDTYROU,DSC$K_DTYPE_BU,K_UNSDESROU,K_UNSDESROU
: 1138      1267 1      ,K_UNSDTYROU,K_UNSDESROU,K_UNSDESROU,K_UNSDESROU,K_UNSDESROU
: 1139      1268 1      ,K_UNSDESROU,K_UNSDESROU,K_UNSDTYROU,K_UNSDTYROU,DSC$K_DTYPE_T
: 1140      1269 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1141      1270 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDTYROU
: 1142      1271 1      ,K_UNSDTYROU,K_UNSDTYROU,K_UNSDESROU,K_UNSDESROU,K_UNSDTYROU
: 1143      1272 1      ,K_UNSDDESSTA,K_UNSDTYROU,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1144      1273 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1145      1274 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1146      1275 1      %( State five. (Class p. )%
: 1147      1276 1      %( State six. (Class 'undefined' )%
: 1148      1277 1      %( State seven. (Class j. )%
: 1149      1278 1      %( State eight. (Class 'undefined' )%
: 1150      1279 1      %( State nine. (Class sd. )%
: 1151      1280 1      ,K_UNSDDESSTA,K_UNSDDESSTA,DSC$K_DTYPE_BU,DSC$K_DTYPE_WU,DSC$K_DTYPE_LU
: 1152      1281 1      ,DSC$K_DTYPE_QD,DSC$K_DTYPE_B,DSC$K_DTYPE_W,DSC$K_DTYPE_L,DSC$K_DTYPE_Q
: 1153      1282 1      ,DSC$K_DTYPE_F,DSC$K_DTYPE_D,DSC$K_DTYPE_FC,DSC$K_DTYPE_DC,DSC$K_DTYPE_T
: 1154      1283 1      ,DSC$K_DTYPE_NU,DSC$K_DTYPE_NL,DSC$K_DTYPE_NLO,DSC$K_DTYPE_NR,DSC$K_DTYPE_NRO
: 1155      1284 1      ,DSC$K_DTYPE_NZ,DSC$K_DTYPE_P,K_UNSDDESSTA,R_UNSDDESSTA,K_UNSDDESSTA
: 1156      1285 1      ,K_UNSDDESSTA,DSC$K_DTYPE_O,DSC$K_DTYPE_G,DSC$K_DTYPE_H,DSC$K_DTYPE_GC
: 1157      1286 1      ,DSC$K_DTYPE_HC,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,R_UNSDDESSTA
: 1158      1287 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1159      1288 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1160      1289 1      %( State ten. (Class nca. )%
: 1161      1290 1      ,K_UNSDTYROU,K_UNSDTYROU,DSC$K_DTYPE_BU,K_UNSDESROU,K_UNSDESROU
: 1162      1291 1      ,K_UNSDTYROU,K_UNSDESROU,K_UNSDESROU,K_UNSDESROU,K_UNSDESROU
: 1163      1292 1      ,K_UNSDESROU,K_UNSDESROU,K_UNSDTYROU,K_UNSDTYROU,DSC$K_DTYPE_T
: 1164      1293 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1165      1294 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDTYROU
: 1166      1295 1      ,K_UNSDTYROU,K_UNSDTYROU,K_UNSDESROU,K_UNSDESROU,K_UNSDTYROU
: 1167      1296 1      ,K_UNSDDESSTA,K_UNSDTYROU,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1168      1297 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1169      1298 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1170      1299 1      %( State eleven. (Class vs. )%
: 1171      1300 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1172      1301 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1173      1302 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,DSC$K_DTYPE_T
: 1174      1303 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1175      1304 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1176      1305 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1177      1306 1      ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1178      1307 1      ,K_UNSDDESSTA,K_UNSDDESSTA,DSC$K_DTYPE_VT,DSC$K_DTYPE_AC
: 1179      1308 1      ,DSC$K_DTYPE_AZ,K_UNSDDESSTA,K_UNSDDESSTA,K_UNSDDESSTA
: 1180      1309 1      %( State twelve. (Class vsa. )%
: 1181      1310 1      %( State thirteen. (Class uba. )%
: 1182      1311 1      ,K_UNSDTYROU,DSC$K_DTYPE_V,DSC$K_DTYPE_BU,DSC$K_DTYPE_WU,DSC$K_DTYPE_LU

```

```
1183 1312 1 .DSCSK_DTYPE_QU,DSCSK_DTYPE_B,DSCSK_DTYPE_W,DSCSK_DTYPE_L,DSCSK_DTYPE_Q
1184 1313 1 .DSCSK_DTYPE_F,DSCSK_DTYPE_D,DSCSK_DTYPE_FC,DSCSK_DTYPE_DC,DSCSK_DTYPE_T
1185 1314 1 .DSCSK_DTYPE_NU,DSCSK_DTYPE_NL,DSCSK_DTYPE_NLO,DSCSK_DTYPE_NR,DSCSK_DTYPE_NRO
1186 1315 1 .DSCSK_DTYPE_NZ,DSCSK_DTYPE_P,K_UNSDTYROU,K_UNSDTYROO,K_UNSDESSTA
1187 1316 1 .K_UNSDTYROU,DSCSK_DTYPE_O,DSCSK_DTYPE_G,DSCSK_DTYPE_H,DSCSK_DTYPE_GC
1188 1317 1 .DSCSK_DTYPE_HC,K_UNSDTYROU,K_UNSDTYROO,K_UNSDTYROU,DSCSK_DTYPE_VU
1189 1318 1 .K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
1190 1319 1 .DSCSK_DTYPE_TF,DSCSK_DTYPE_SV,DSCSK_DTYPE_SVU
1191 1320 1 % ( State fourteen. Class uba. )%
1192 1321 1 % ( Add more states below )%
1193 1322 1 ) : STATES;
1194 1323 1
1195 1324 1
1196 1325 1 ! Final States.
1197 1326 1
1198 1327 1 ! These are the final states that are valid CLASS, DATA TYPE combinations.
1199 1328 1 ! The rest of the final states are error states.
1200 1329 1 ! The first argument to the macro is CLASS, and the second is the DATA TYPE.
1201 1330 1
1202 1331 1 LITERAL
1203 1332 1 K_S_BU = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_BU),
1204 1333 1 K_S_WU = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_WU),
1205 1334 1 K_S_LU = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_LU),
1206 1335 1 K_S_B = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_B),
1207 1336 1 K_S_W = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_W),
1208 1337 1 K_S_L = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_L),
1209 1338 1 K_S_V = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_V),
1210 1339 1 K_S_SV = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_SV),
1211 1340 1 K_S_TF = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_TF),
1212 1341 1 K_S_Q = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_Q),
1213 1342 1 K_S_QU = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_QU),
1214 1343 1 K_S_O = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_O),
1215 1344 1 K_S_F = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_F),
1216 1345 1 K_S_FC = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_FC),
1217 1346 1 K_S_D = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_D),
1218 1347 1 K_S_DC = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_DC),
1219 1348 1 K_S_T = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_T),
1220 1349 1 K_S_NU = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_NU),
1221 1350 1 K_S_NL = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_NL),
1222 1351 1 K_S_NLO = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_NLO),
1223 1352 1 K_S_NR = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_NR),
1224 1353 1 K_S_NRO = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_NRO),
1225 1354 1 K_S_NZ = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_NZ),
1226 1355 1 K_S_ZI = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_ZI),
1227 1356 1 K_S_P = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_P),
1228 1357 1 K_S_G = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_G),
1229 1358 1 K_S_GC = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_GC),
1230 1359 1 K_S_H = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_H),
1231 1360 1 K_S_HC = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_HC),
1232 1361 1 K_UBS_V = FINAL_STATE (DSCSK_CLASS_UBS, DSCSK_DTYPE_V),
1233 1362 1 K_UBS_BU = FINAL_STATE (DSCSK_CLASS_UBS, DSCSK_DTYPE_BU),
1234 1363 1 K_UBS_WU = FINAL_STATE (DSCSK_CLASS_UBS, DSCSK_DTYPE_WU),
1235 1364 1 K_UBS_LU = FINAL_STATE (DSCSK_CLASS_UBS, DSCSK_DTYPE_LU),
1236 1365 1 K_UBS_B = FINAL_STATE (DSCSK_CLASS_UBS, DSCSK_DTYPE_B),
1237 1366 1 K_UBS_W = FINAL_STATE (DSCSK_CLASS_UBS, DSCSK_DTYPE_W),
1238 1367 1 K_UBS_L = FINAL_STATE (DSCSK_CLASS_UBS, DSCSK_DTYPE_L),
1239 1368 1 K_UBS_Q = FINAL_STATE (DSCSK_CLASS_UBS, DSCSK_DTYPE_Q),
```

```

1240 1369 1 K_UBS_QU = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_QU),
1241 1370 1 K_UBS_F = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_F),
1242 1371 1 K_UBS_D = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_D),
1243 1372 1 K_UBS_FC = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_FC),
1244 1373 1 K_UBS_DC = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_DC),
1245 1374 1 K_UBS_T = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_T),
1246 1375 1 K_UBS_NU = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_NU),
1247 1376 1 K_UBS_NL = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_NL),
1248 1377 1 K_UBS_NLO = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_NLO),
1249 1378 1 K_UBS_NR = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_NR),
1250 1379 1 K_UBS_NRO = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_NRO),
1251 1380 1 K_UBS_NZ = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_NZ),
1252 1381 1 K_UBS_P = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_P),
1253 1382 1 K_UBS_O = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_O),
1254 1383 1 K_UBS_G = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_G),
1255 1384 1 K_UBS_H = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_H),
1256 1385 1 K_UBS_GC = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_GC),
1257 1386 1 K_UBS_HC = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_HC),
1258 1387 1 K_UBS_SV = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_SV),
1259 1388 1 K_UBS_VU = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_VU),
1260 1389 1 K_UBS_SVU = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_SVU),
1261 1390 1 K_UBS_TF = FINAL_STATE (DSC$K_CLASS_UBS, DSC$K_DTYPE_TF),
1262 1391 1 K_D_T = FINAL_STATE (DSC$K_CLASS_D, DSC$K_DTYPE_T),
1263 1392 1 K_A_BU = FINAL_STATE (DSC$K_CLASS_A, DSC$K_DTYPE_BU),
1264 1393 1 K_A_T = FINAL_STATE (DSC$K_CLASS_A, DSC$K_DTYPE_T),
1265 1394 1 K_SD_BU = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_BU),
1266 1395 1 K_SD_WU = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_WU),
1267 1396 1 K_SD_LU = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_LU),
1268 1397 1 K_SD_B = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_B),
1269 1398 1 K_SD_W = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_W),
1270 1399 1 K_SD_L = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_L),
1271 1400 1 K_SD_Q = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_Q),
1272 1401 1 K_SD_QU = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_QU),
1273 1402 1 K_SD_O = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_O),
1274 1403 1 K_SD_F = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_F),
1275 1404 1 K_SD_FC = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_FC),
1276 1405 1 K_SD_D = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_D),
1277 1406 1 K_SD_DC = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_DC),
1278 1407 1 K_SD_G = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_G),
1279 1408 1 K_SD_GC = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_GC),
1280 1409 1 K_SD_H = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_H),
1281 1410 1 K_SD_HC = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_HC),
1282 1411 1 K_SD_T = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_T),
1283 1412 1 K_SD_NU = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_NU),
1284 1413 1 K_SD_NL = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_NL),
1285 1414 1 K_SD_NLO = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_NLO),
1286 1415 1 K_SD_NR = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_NR),
1287 1416 1 K_SD_NRO = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_NRO),
1288 1417 1 K_SD_NZ = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_NZ),
1289 1418 1 K_SD_P = FINAL_STATE (DSC$K_CLASS_SD, DSC$K_DTYPE_P),
1290 1419 1 K_NCA_BU = FINAL_STATE (DSC$K_CLASS_NCA, DSC$K_DTYPE_BU),
1291 1420 1 K_NCA_T = FINAL_STATE (DSC$K_CLASS_NCA, DSC$K_DTYPE_T),
1292 1421 1 K_VS_AC = FINAL_STATE (DSC$K_CLASS_VS, DSC$K_DTYPE_AC),
1293 1422 1 K_VS_AZ = FINAL_STATE (DSC$K_CLASS_VS, DSC$K_DTYPE_AZ),
1294 1423 1 K_VS_T = FINAL_STATE (DSC$K_CLASS_VS, DSC$K_DTYPE_T),
1295 1424 1 K_VS_VT = FINAL_STATE (DSC$K_CLASS_VS, DSC$K_DTYPE_VT),
1296 1425 1 K_SMEFINSTA = FINAL_STATE (DSC$K_CLASS_S, DSC$K_DTYPE_V),

```

! Smallest final state supported.

DBGCVTDX  
V04-000

<sup>4</sup>  
15-Sep-1984 23:57:30  
14-Sep-1984 12:16:44

VAX-11 Bliss-32 V4.0-742  
[DEBUG.SRC]DBGCVTDX.B32;1

Page 25  
(5)

:	1297	1426	1	K_LRGFINSTA = FINAL_STATE (DSC\$K_CLASS_UBS, DSC\$K_DTYPE_SVU),	! Largest final state supported.
:	1298	1427	1	K_TOP_SD = FINAL_STATE (DSC\$K_CLASS_SD, DSC\$K_DTYPE_HC);	! Top state for class SD.
:	1299	1428	1	K_BOTTOM_SD = FINAL_STATE (DSC\$K_CLASS_SD, DSC\$K_DTYPE_B);	! Bottom state for class SD.

```

1301 1429 1 GLOBAL ROUTINE DBG$COVER_DX_DX (SRC_VALUE_DESC, DST_VALUE_DESC, CVT_ROUND_FLAG) =
1302 1430 1
1303 1431 1 FUNCTION
1304 1432 1     This routine is a cover function for DBG$CVT_DX_DX. It has
1305 1433 1     two purposes:
1306 1434 1     1. To declare a handler which screens errors and changes them to
1307 1435 1     the appropriate DEBUG error.
1308 1436 1     2. To dummy in the correct class for DBG$CVT_DX_DX
1309 1437 1
1310 1438 1 INPUTS
1311 1439 1     SRC_VALUE_DESC - Pointer to a value descriptor to be type-converted.
1312 1440 1
1313 1441 1     DST_VALUE_DESC - Pointer to the target value descriptor.
1314 1442 1
1315 1443 1     CVT_ROUND_FLAG - A flag set to TRUE to indicate the rounding takes
1316 1444 1     place in conversion.
1317 1445 1
1318 1446 1 OUTPUTS
1319 1447 1     A pointer to a value descriptor is returned. The target descriptor
1320 1448 1     is filled in with the result of the conversion.
1321 1449 1
1322 1450 2 BEGIN
1323 1451 2
1324 1452 2 MAP
1325 1453 2     SRC_VALUE_DESC: REF DBG$VALDESC,
1326 1454 2     DST_VALUE_DESC: REF DBG$VALDESC;
1327 1455 2
1328 1456 2 LOCAL
1329 1457 2     DUMMY,           ! A dummy parameter.
1330 1458 2     FCODE,           ! Fcode for the data object
1331 1459 2     STATUS,          ! Return status from typeid check.
1332 1460 2     SOURCE_CLASS: BYTE, ! Class of Source VMS descriptor
1333 1461 2     TARGET_CLASS: BYTE, ! Class of Target VMS descriptor
1334 1462 2     SOURCE_DTYPE: BYTE, ! Dtype of Source VMS descriptor
1335 1463 2     TARGET_DTYPE: BYTE, ! Dtype of Target VMS descriptor
1336 1464 2     SOURCE_LENGTH: WORD, ! Length of Source VMS descriptor
1337 1465 2     TARGET_LENGTH: WORD, ! Length of Target VMS descriptor
1338 1466 2     DESC_VAL: REF DBG$VALDESC, ! Pointer to source or target value descriptor
1339 1467 2     DESC_PTR: REF BLOCK[,BYTE], ! Pointer to source or target descriptor.
1340 1468 2     SOURCE: REF BLOCK[,BYTE], ! Address of VMS descriptor
1341 1469 2     TARGET: REF BLOCK[,BYTE], ! Address of VMS descriptor
1342 1470 2     TYPEID_INDEX;         ! Typeid index to perform the typeid
1343 1471 2                             check
1344 1472 2
1345 1473 2
1346 1474 2 ! Recover the VMS descriptors.
1347 1475 2
1348 1476 2 SOURCE = SRC_VALUE_DESC[DBG$A_VALUE_VMSDESC];
1349 1477 2 TARGET = DST_VALUE_DESC[DBG$A_VALUE_VMSDESC];
1350 1478 2
1351 1479 2
1352 1480 2 ! Save pointer to result.
1353 1481 2
1354 1482 2 SAVE_RESULT = .DST_VALUE_DESC[DBG$L_VALUE_POINTER];
1355 1483 2
1356 1484 2
1357 1485 2 ! Dummy in the correct class field. (first saving away the old ones.)

```



```
1358 1486 2
1359 1487 2
1360 1488 2
1361 1489 2
1362 1490 2
1363 1491 2
1364 1492 2
1365 1493 2
1366 1494 2
1367 1495 2
1368 1496 2
1369 1497 2
1370 1498 2
1371 1499 2
1372 1500 2
1373 1501 2
1374 1502 2
1375 1503 2
1376 1504 2
1377 1505 2
1378 1506 2
1379 1507 2
1380 1508 2
1381 1509 2
1382 1510 2
1383 1511 2
1384 1512 2
1385 1513 2
1386 1514 2
1387 1515 2
1388 1516 2
1389 1517 2
1390 1518 2
1391 1519 2
1392 1520 2
1393 1521 2
1394 1522 2
1395 1523 2
1396 1524 2
1397 1525 2
1398 1526 2
1399 1527 2
1400 1528 2
1401 1529 2
1402 1530 2
1403 1531 2
1404 1532 2
1405 1533 2
1406 1534 2
1407 1535 2
1408 1536 2
1409 1537 2
1410 1538 2
1411 1539 2
1412 1540 2
1413 1541 2
1414 1542 2

!
SOURCE_CLASS = .SOURCE[DSC$B_CLASS];
TARGET_CLASS = .TARGET[DSC$B_CLASS];
SOURCE_DTYPE = .SOURCE[DSC$B_DTYPE];
TARGET_DTYPE = .TARGET[DSC$B_DTYPE];
SOURCE_LENGTH = .SOURCE[DSC$B_LENGTH];

! The debugger doesn't handle dynamic string descriptors. Some output is
! better than none, so we treat them as regular string descriptors, and
! truncate/pad as required.
!
IF .SOURCE_CLASS EQL DSC$K_CLASS_D AND .SOURCE[DSC$B_DTYPE] EQL DSC$K_DTYPE_T
THEN
    SOURCE[DSC$B_CLASS] = DSC$K_CLASS_S;
IF .TARGET_CLASS EQL DSC$K_CLASS_D AND .TARGET[DSC$B_DTYPE] EQL DSC$K_DTYPE_T
THEN
    TARGET[DSC$B_CLASS] = DSC$K_CLASS_S;

! If class field is zero, map in correct class/dtype.
!
IF .SOURCE[DSC$B_CLASS] EQL 0
THEN
    SOURCE[DSC$B_CLASS] = DBG$MAP_DTYPE_CLASS(.SOURCE[DSC$B_DTYPE], FALSE);
IF .TARGET[DSC$B_CLASS] EQL 0
THEN
    TARGET[DSC$B_CLASS] = DBG$MAP_DTYPE_CLASS(.TARGET[DSC$B_DTYPE], FALSE);

! Case on the fcode. If the target data is one of the non-standard
! data types then typeid and/or range value will be validated by
! calling DBG$PERFORM_TYPEID_CHECK. First set up the routine check
! index according to fcode.
!
FCODE = .DST VALUE DESC[DBG$B_DHDR_FCODE];
CASE .FCODE FROM RST$K_TYPE_MINIMUM TO RST$K_TYPE_MAXIMUM OF
SET
    [RST$K_TYPE_ENUM]:
        TYPEID_INDEX = ORT$K_TYPEID_ENUM_ENUM;

    [RST$K_TYPE_SET]:
        TYPEID_INDEX = ORT$K_TYPEID_SET_SET;

    [RST$K_TYPE_SUBRNG]:
        TYPEID_INDEX = ORT$K_TYPEID_SUBRNG_SUBRNG;

    [INRANGE, OTRANGE]:
        TYPEID_INDEX = 0;

TES;

! If routine check index is set up, call dbg$perform_typeid_check
! to perform the typeid check.
```

```

: 1415      1543  2  IF .TYPEID_INDEX NEQ 0
: 1416      1544  2  THEN
: 1417      1545  3      BEGIN
: 1418      1546  3          STATUS = DBG$PERFORM_TYPEID_CHECK(.TYPEID_INDEX,
: 1419      1547  3              .SRC_VALUE_DESC, .DST_VALUE_DESC, 0);
: 1420      1548  3
: 1421      1549  3          IF NOT .STATUS THEN SIGNAL(DBG$_OPNOTALLOW, 1, .DBG$GL_OPCODE_NAME);
: 1422      1550  2      END;
: 1423      1551  2
: 1424      1552  2
: 1425      1553  2      ! Now, typeid has checked, deposit is legal operation for both
: 1426      1554  2      ! standard and non-standard data types at this point.
: 1427      1555  2      ! Fixup the class and dtype fields to be vax standard format, so
: 1428      1556  2      ! DBG$CVT_DX_DX can be called to perform the conversion.
: 1429      1557  2
: 1430      1558  2  INCR I FROM 0 TO 1 DO
: 1431      1559  3      BEGIN
: 1432      1560  3          IF .I EQL 0
: 1433      1561  3          THEN
: 1434      1562  4              BEGIN
: 1435      1563  4                  DESC_VAL = .SRC_VALUE_DESC;
: 1436      1564  4                  DESC_PTR = .SOURCE;
: 1437      1565  4              END
: 1438      1566  3          ELSE
: 1439      1567  4              BEGIN
: 1440      1568  4                  DESC_VAL = .DST_VALUE_DESC;
: 1441      1569  4                  DESC_PTR = .TARGET;
: 1442      1570  3              END;
: 1443      1571  3
: 1444      1572  4              IF (.DESC_VAL[DBG$B_VALUE_DTYPE] EQL 0 AND
: 1445      1573  4                  .DESC_VAL[DBG$B_VALUE_CLASS] EQL 0)
: 1446      1574  3              THEN
: 1447      1575  3                  DESC_PTR = COVER_VMSDESC_SETUP(.DESC_VAL[DBG$L_DHDR_TYPEID],
: 1448      1576  3                      .DESC_PTR);
: 1449      1577  2              END;
: 1450      1578  2
: 1451      1579  2
: 1452      1580  2      ! Adjust the length of the source. So we won't get truncation message.
: 1453      1581  2      ! This is used for, ie., DEP enum=1, where enum is allocated 1 byte, and
: 1454      1582  2      ! 1 is 1 longword. in some cases, we'll get integer overflow message.
: 1455      1583  2
: 1456      1584  2  SELECTONE .FCODE OF
: 1457      1585  2      SET
: 1458      1586  2          [RST$K_TYPE_ENUM, RST$K_TYPE_SUBRNG]:
: 1459      1587  3      BEGIN
: 1460      1588  3          IF .SRC_VALUE_DESC[DBG$L_DHDR_TYPEID] EQL 0
: 1461      1589  3          THEN
: 1462      1590  4              BEGIN
: 1463      1591  4                  IF .SRC_VALUE_DESC[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_ATOMIC
: 1464      1592  4                  THEN
: 1465      1593  5                      BEGIN
: 1466      1594  5                          SOURCE[DSC$B_CLASS] = .TARGET[DSC$B_CLASS];
: 1467      1595  5                          SOURCE[DSC$B_DTYPE] = .TARGET[DSC$B_DTYPE];
: 1468      1596  5                          SOURCE[DSC$W_LENGTH] = .TARGET[DSC$W_LENGTH];
: 1469      1597  4                      END;
: 1470      1598  4
: 1471      1599  3      END;
```

```

: 1472 1600 3
: 1473 1601 2
: 1474 1602 2
: 1475 1603 2
: 1476 1604 2
: 1477 1605 2
: 1478 1606 2
: 1479 1607 2
: 1480 1608 2
: 1481 1609 2
: 1482 1610 2
: 1483 1611 2
: 1484 1612 2
: 1485 1613 2
: 1486 1614 2
: 1487 1615 2
: 1488 1616 2
: 1489 1617 3
: 1490 1618 3
: 1491 1619 3
: 1492 1620 3
: 1493 1621 3
: 1494 1622 3
: 1495 1623 3
: 1496 1624 4
: 1497 1625 4
: 1498 1626 4
: 1499 1627 4
: 1500 1628 4
: 1501 1629 3
: 1502 1630 4
: 1503 1631 4
: 1504 1632 4
: 1505 1633 4
: 1506 1634 5
: 1507 1635 4
: 1508 1636 4
: 1509 1637 4
: 1510 1638 4
: 1511 1639 3
: 1512 1640 2
: 1513 1641 2
: 1514 1642 2
: 1515 1643 2
: 1516 1644 2
: 1517 1645 2
: 1518 1646 2
: 1519 1647 2
: 1520 1648 2
: 1521 1649 2
: 1522 1650 2
: 1523 1651 2
: 1524 1652 2
: 1525 1653 2
: 1526 1654 2
: 1527 1655 2
: 1528 1656 2

```

```

        END;
    [OTHERWISE]:
        0;
    TES;

! Do the conversion. Put everything back.
!
SELECTONE .FCODE OF
    SET
    [RST$K_TYPE_RFA]:
        CH$MOVE(.DST_VALUE_DESC[DBG$W_VALUE_LENGTH],
            .SRC_VALUE_DESC[DBG$L_VALUE_POINTER], .DST_VALUE_DESC[DBG$L_VALUE_POINTER]);
    [RST$K_TYPE_SET]:
        BEGIN
        LOCAL
            INDEX,
            SETVALUE: REF BITVECTOR[];

        IF .SRC_VALUE_DESC[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_SET
        THEN
            BEGIN
            CH$MOVE(.DST_VALUE_DESC[DBG$W_VALUE_LENGTH],
                .SRC_VALUE_DESC[DBG$L_VALUE_POINTER], .DST_VALUE_DESC[DBG$L_VALUE_POINTER]);
            END
        ELSE
            BEGIN
            INDEX = ..SRC_VALUE_DESC[DBG$L_VALUE_POINTER];
            IF .INDEX LSS 0 THEN SIGNAL(DBG$B_BITRANGE);
            SETVALUE = .DST_VALUE_DESC[DBG$L_VALUE_POINTER];
            IF .INDEX LEQ (.DST_VALUE_DESC[DBG$W_VALUE_LENGTH] * 8 - 1)
            THEN
                SETVALUE[.INDEX] = 1
            ELSE
                SIGNAL(DBG$B_BITRANGE);
            END;
        END;
    END;

    [OTHERWISE]:
        DBG$CVT_DX_DX (.SOURCE, .TARGET, DUMMY, .CVT_ROUND_FLAG);
    TES;

SOURCE[DSC$B_CLASS] = .SOURCE_CLASS;
SOURCE[DSC$B_DTYPE] = .SOURCE_DTYPE;
SOURCE[DSC$W_LENGTH] = .SOURCE_LENGTH;
TARGET[DSC$B_CLASS] = .TARGET_CLASS;
TARGET[DSC$B_DTYPE] = .TARGET_DTYPE;

! Do range check.
!
IF .TYPEID_INDEX NEQ 0

```

```

: 1529      1657 2      THEN
: 1530      1658 3      BEGIN
: 1531      1659 3      STATUS = DBG$PERFORM_TYPEID_CHECK (.TYPEID_INDEX,
: 1532      1660 3      .SRC_VALUE_DESC, 0, .DST_VALUE_DESC);
: 1533      1661 3      IF NOT .STATUS
: 1534      1662 3      THEN
: 1535      1663 3      SIGNAL (DBG$_IVALOUTBND, 1, .DBG$GL_OPCODE_NAME);
: 1536      1664 2      END;
: 1537      1665 2      RETURN .DST_VALUE_DESC;
: 1538      1666 2      END;
: 1539      1667 1
```

																.TITLE	DBGCVTDX																
																.IDENT	\V04-000\																
																.PSECT	DBG\$PLIT,NOWRT,	SHR,	PIC,0														
FF	0D	FF	0B	0A	09	FB	FF	FB	FF	04	FF	02	01	FF	00000	P.AAB:	.BYTE	-1,	1,	2,	-1,	4,	-1,	-5,	-1,	-5,	9,	10,	-	:			
OE	0D	0C	0B	0A	09	08	07	06	05	04	03	02	01	FE	0000F	P.AAC:	.BYTE	11,	-1,	13,	-1,										:		
1D	1C	1B	1A	FE	FC	FE	16	15	14	13	12	11	10	0F	0001E			-2,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	-	:		
FC	FC	FE	29	28	FC	FC	FC	FC	FC	FE	FE	FE	FE	1E	0002D			12,	13,	14,	15,	16,	17,	18,	19,	20,	21,	-	-	:			
FC	FC	0E	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	0003C			-2,	-2,	-2,	-4,	-4,	-4,	-4,	-4,	-4,	40,	41,	-	-	:		
FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	0004B			-2,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-	-	:	
FD	02	FE	FE	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	0005A			-4,	-4,	-4,	-4,	-4,	14,	-4,	-4,	-4,	-4,	-4,	-4,	-	-	:	
FC	FC	FC	FC	0E	FE	FE	FD	FD	FD	FD	FD	FD	FE	FD	00069			-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-	-	:	
FC	FC	FE	FC	FE	FD	FD	FE	FE	FE	FC	FC	FC	FC	FC	00078			-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-	-	:	
05	04	03	02	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	00087			-4,	-4,	-4,	-4,	-2,	-2,	2,	-3,	-3,	-2,	-	-	-	:		
14	13	12	11	10	0F	0E	0D	0C	0B	0A	09	08	07	06	00096			-3,	-3,	-3,	-3,	-3,	-3,	-2,	-2,	14,	-4,	-4,	-	-	:		
FC	FC	FC	FC	FC	1E	1D	1C	1B	1A	FC	FC	FC	FC	15	000A5			-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-2,	-2,	-	-	-	:		
FD	FD	FE	FD	FD	02	FE	FE	FC	FC	FC	FC	FC	FC	FC	000B4			-2,	-3,	-3,	-2,	-4,	-2,	-4,	-4,	-4,	-4,	-4,	-	-	:		
FC	FC	FC	FC	FC	FC	FC	FC	0E	FE	FE	FD	FD	FD	FD	000C3			-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	2,	3,	-	-	:	
FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	000D2			4,	5,	6,	7,	8,	9,	10,	11,	12,	13,	14,	15,	-	:		
FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	000E1			16,	17,	18,	19,	20,	21,	-4,	-4,	-4,	-4,	-	-	:			
FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	000F0			26,	27,	28,	29,	30,	-4,	-4,	-4,	-4,	-4,	-	-	:			
27	26	25	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	FC	000FF			-4,	-4,	-4,	-4,	-4,	-4,	-2,	-2,	2,	-	-	-	:			
0B	0A	09	08	07	06	05	04	03	02	01	FE	FC	FC	FC	0010E			-3,	-3,	-2,	-3,	-3,	-3,	-3,	-3,	-3,	-2,	-	-	:			
1A	FE	FC	FE	FE	15	14	13	12	11	10	0F	0E	0D	0C	0011D			-2,	14,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-	-	:			
29	28	FC	FC	FC	FC	FC	22	FE	FE	FE	1E	1D	1C	1B	0012C			-4,	-2,	-2,	-2,	-3,	-3,	-2,	-4,	-2,	-4,	-	-	:			
														2A	0013B			-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-	-	:		
																		-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-	-	:		
																		-4,	-4,	-4,	-4,	14,	-4,	-4,	-4,	-4,	-4,	-4,	-	-	:		
																		-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-4,	-	-	:		
																		-4,	-4,	-4,	-4,	-4,	-4,	-4,	37,	38,	39,	-	-	:			
																		-4,	-4,	-4,	-2,	1,	2,	3,	4,	5,	6,	7,	8,	-	:		
																		9,	10,	11,	12,	13,	14,	15,	16,	17,	18,	-	-	:			
																		19,	20,	21,	-2,	-2,	-4,	-2,	26,	27,	28,	-	-	:			
																		29,	30,	-2,	-2,	-2,	34,	-4,	-4,	-4,	-4,	-	-	:			
																		-4,	40,	41,	42,									:			
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	0013C	P.AAD:	.ASCII	<24>\DBGCVTDX:	invalid class\							:							
					73	73	61	6C	63	20	64	69	6C	61	0014B															:			
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	00155	P.AAE:	.ASCII	<24>\DBGCVTDX:	invalid class\							:							
					73	73	61	6C	63	20	64	69	6C	61	00164															:			
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	0016E	P.AAF:	.ASCII	<24>\DBGCVTDX:	invalid class\							:							
					73	73	61	6C	63	20	64	69	6C	61	0017D															:			
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	00187	P.AAG:	.ASCII	<24>\DBGCVTDX:	invalid class\							:							



76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	0045C	P.ABJ:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	0046B				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	00475	P.ABK:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	00484				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	0048E	P.ABL:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	0049D				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	004A7	P.ABM:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	004B6				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	004C0	P.ABN:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	004CF				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	004D9	P.ABO:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	004E8				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	004F2	P.ABP:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	00501				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	0050B	P.ABQ:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	0051A				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	00524	P.ABR:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	00533				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	0053D	P.ABS:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	0054C				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	00556	P.ABT:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	00565				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	0056F	P.ABU:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	0057E				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	00588	P.ABV:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	00597				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	005A1	P.ABW:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	005B0				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	005BA	P.ABX:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	005C9				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	005D3	P.ABY:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	005E2				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	005EC	P.ABZ:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	005FB				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	00605	P.ACA:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	00614				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	0061E	P.ACB:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	0062D				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	00637	P.ACC:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	00646				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	00650	P.ACD:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	0065F				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	00669	P.ACE:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	00678				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	00682	P.ACF:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	00691				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	0069B	P.ACG:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	006AA				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	006B4	P.ACH:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	006C3				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	006CD	P.ACI:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	006DC				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	006E6	P.ACJ:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	006F5				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	006FF	P.ACK:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	0070E				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	00718	P.ACL:	.ASCII	<24>\DBGCVTDX:	invalid class\

```

E 5
15-Sep-1984 23:57:30 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:16:44 [DEBUG.SRC]DBGCVTDX.B32;1

```

76	6E	69	20	20	73	73	61	6C	63	20	64	69	6C	61	00727				
					3A	58	44	54	56	43	47	42	44	18	00731	P.ACM:	.ASCII	<24>\DBGCVTDX:	invalid class\
76	6E	69	20	20	73	73	61	6C	63	20	64	69	6C	61	00740				
					3A	58	44	54	56	43	47	42	44	18	0074A	P.AC�:	.ASCII	<24>\DBGCVTDX:	invalid class\
76	6E	69	20	20	73	73	61	6C	63	20	64	69	6C	61	00759				
					3A	58	44	54	56	43	47	42	44	18	00763	P.ACO:	.ASCII	<24>\DBGCVTDX:	invalid class\
76	6E	69	20	20	73	73	61	6C	63	20	64	69	6C	61	00772				
					3A	58	44	54	56	43	47	42	44	18	0077C	P.ACP:	.ASCII	<24>\DBGCVTDX:	invalid class\
76	6E	69	20	20	73	73	61	6C	63	20	64	69	6C	61	00788				
					3A	58	44	54	56	43	47	42	44	18	00795	P.ACQ:	.ASCII	<24>\DBGCVTDX:	invalid class\
76	6E	69	20	20	73	73	61	6C	63	20	64	69	6C	61	007A4				
					3A	58	44	54	56	43	47	42	44	18	007AE	P.ACR:	.ASCII	<24>\DBGCVTDX:	invalid class\
76	6E	69	20	20	73	73	61	6C	63	20	64	69	6C	61	007BD				
					3A	58	44	54	56	43	47	42	44	18	007C7	P.ACS:	.ASCII	<24>\DBGCVTDX:	invalid class\
76	6E	69	20	20	73	73	61	6C	63	20	64	69	6C	61	007D6				
					3A	58	44	54	56	43	47	42	44	18	007E0	P.ACT:	.ASCII	<24>\DBGCVTDX:	invalid class\
76	6E	69	20	20	73	73	61	6C	63	20	64	69	6C	61	007EF				
					3A	58	44	54	56	43	47	42	44	18	007F9	P.ACU:	.ASCII	<24>\DBGCVTDX:	invalid class\
76	6E	69	20	20	73	73	61	6C	63	20	64	69	6C	61	00808				
					3A	58	44	54	56	43	47	42	44	18	00812	P.ACV:	.ASCII	<24>\DBGCVTDX:	invalid class\
76	6E	69	20	20	73	73	61	6C	63	20	64	69	6C	61	00821				
					3A	58	44	54	56	43	47	42	44	18	0082B	P.ACW:	.ASCII	<24>\DBGCVTDX:	invalid class\
76	6E	69	20	20	73	73	61	6C	63	20	64	69	6C	61	0083A				
					3A	58	44	54	56	43	47	42	44	18	00844	P.ACX:	.ASCII	<24>\DBGCVTDX:	invalid class\
76	6E	69	20	20	73	73	61	6C	63	20	64	69	6C	61	00853				
					3A	58	44	54	56	43	47	42	44	18	0085D	P.ACY:	.ASCII	<24>\DBGCVTDX:	invalid class\
76	6E	69	20	20	73	73	61	6C	63	20	64	69	6C	61	0086C				
					3A	58	44	54	56	43	47	42	44	18	00876	P.ACZ:	.ASCII	<24>\DBGCVTDX:	invalid class\
76	6E	69	20	20	73	73	61	6C	63	20	64	69	6C	61	00885				
					3A	58	44	54	56	43	47	42							

DBGCVTDX  
V04-000

F 5  
15-Sep-1984 23:57:30 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:16:44 [DEBUG.SRC]DBGCVTDX.B32;1

Page 34  
(6)

76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	009ED	P.ADO:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	009FC				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	00A06	P.ADP:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	00A15				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	00A1F	P.ADQ:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	00A2E				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	00A38	P.ADR:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	00A47				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	00A51	P.ADS:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	00A60				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	00A6A	P.ADT:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	00A79				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	00A83	P.ADU:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	00A92				
76	6E	69	20	20	3A	58	44	54	56	43	47	42	44	18	00A9C	P.ADV:	.ASCII	<24>\DBGCVTDX:	invalid class\
					73	73	61	6C	63	20	64	69	6C	61	00AAB				

.PSECT DBG\$OWN,NOEXE, PIC,2

00000 DECIMAL\_CONVERT:

.BLKB 4

00004 SAVE\_RESULT:

.BLKB 4

CLASS\_TABLE=  
DTYPE\_TABLE=

P.AAB  
P.AAC

.EXTRN DBG\$CVT\_ASHP\_R1  
.EXTRN DBG\$CVT\_CMPH\_R1  
.EXTRN DBG\$CVT\_CVTDH\_R1  
.EXTRN DBG\$CVT\_CVTLB\_R1  
.EXTRN DBG\$CVT\_CVTLH\_R1  
.EXTRN DBG\$CVT\_CVTLW\_R1  
.EXTRN DBG\$CVT\_CVTRD\_R1  
.EXTRN DBG\$CVT\_CVTHD\_R1  
.EXTRN DBG\$CVT\_CVTHF\_R1  
.EXTRN DBG\$CVT\_CVTHG\_R1  
.EXTRN DBG\$CVT\_CVTGH\_R1  
.EXTRN DBG\$CVT\_CVTRHC\_R1  
.EXTRN DBG\$CVT\_CVTRHO\_R1  
.EXTRN DBG\$CVT\_CVTRHQ\_R1  
.EXTRN DBG\$CVT\_CVTRQD\_R1  
.EXTRN DBG\$CVT\_CVTRQH\_R1  
.EXTRN DBG\$CVT\_DIVD2\_R1  
.EXTRN DBG\$CVT\_DIVH2\_R1  
.EXTRN DBG\$CVT\_DIVP\_R1  
.EXTRN DBG\$CVT\_MULD2\_R1  
.EXTRN DBG\$CVT\_MULH2\_R1  
.EXTRN DBG\$CVT\_MULP\_R1  
.EXTRN DBG\$GET\_SET\_TYPEID  
.EXTRN DBG\$INS\_ENCODE, DBG\$MAP\_DTYPE\_CLASS  
.EXTRN DBG\$PERFORM\_TYPEID\_CHECK  
.EXTRN DBG\$STA\_TYP\_SUBRNG  
.EXTRN DBG\$STA\_TYP\_ATOMIC  
.EXTRN DBG\$STRIP\_ZEROES  
.EXTRN FOR\$CVT\_D\_TE, FOR\$CVT\_D\_TF  
.EXTRN FOR\$CVT\_G\_TE, FOR\$CVT\_G\_TF  
.EXTRN FOR\$CVT\_H\_TE, FOR\$CVT\_H\_TF



					.EXTRN	DBG\$CVT_SCALE_OU_UP_BY_10_R1	
					.EXTRN	DBG\$CVT_SCALE_OU_DOWN_BY_10_R1	
					.EXTRN	LIB\$CVT_SCALE_OU_UP_BY_10_R1	
					.EXTRN	LIB\$CVT_SCALE_OU_DOWN_BY_10_R1	
					.EXTRN	DBG\$CVT_SCALE_OU_OP_BY_2_R1	
					.EXTRN	DBG\$CVT_SCALE_OU_DOWN_BY_2_R1	
					.EXTRN	LIB\$MATCH_COND, LIB\$SIG_TO_RET	
					.EXTRN	LIB\$SCOPY_R_DX6	
					.EXTRN	LIB\$SCOPY_DXDX6	
					.EXTRN	LIB\$STOP, MTH\$CVT_D_G	
					.EXTRN	OT\$CVT_L_T1, OT\$CVT_T_D	
					.EXTRN	OT\$CVT_T_G, OT\$CVT_T_R	
					.EXTRN	SY\$ASCTIM, SY\$BINTIM	
					.EXTRN	LIB\$AB_CVTTP_U, LIB\$AB_CVT_O_U	
					.EXTRN	LIB\$AB_CVTTP_O, LIB\$AB_CVT_U_O	
					.EXTRN	LIB\$AB_CVTPT_U, LIB\$AB_CVTPT_O	
					.EXTRN	LIB\$AB_CVTPT_Z, LIB\$AB_CVTTP_Z	
					.EXTRN	DBG\$GL_OPCODE_NAME	
					.EXTRN	LIB\$_STRTRU	
					.PSECT	DBG\$CODE, NOWRT, SHR, PIC, 0	
			OFFC 00000		.ENTRY	DBG\$COVER_DX_DX, Save R2,R3,R4,R5,R6,R7,R8,-; R9,R10,R11	1429
					SUBL2	#36, SP	
					MOVL	SRC_VALUE_DESC, R8	1476
					MOVAB	20(R8), SOURCE	
					MOVL	DST_VALUE_DESC, R7	1477
					MOVAB	20(R7), TARGET	
					MOVL	24(R7), SAVE_RESULT	1482
					MOVAB	3(SOURCE), R11	1487
					MOVB	(R11), SOURCE_CLASS	
					MOVAB	3(TARGET), R10	1488
					MOVB	(R10), TARGET_CLASS	
					MOVB	2(SOURCE), SOURCE_DTYPE	1489
					MOVB	2(TARGET), TARGET_DTYPE	1490
					MOVW	(SOURCE), SOURCE_LENGTH	1491
					CMPB	SOURCE_CLASS, #2	1498
					BNEQ	1\$	
					CMPB	2(SOURCE), #14	
					BNEQ	1\$	
					MOVB	#1, (R11)	1500
					CMPB	TARGET_CLASS, #2	1501
					BNEQ	2\$	
					CMPB	2(TARGET), #14	
					BNEQ	2\$	
					MOVB	#1, (R10)	1503
					TSTB	(R11)	1508
					BNEQ	3\$	
					CLRL	-(SP)	1510
					MOVZBL	2(SOURCE), -(SP)	
					CALLS	#2, DBG\$MAP_DTYPE_CLASS	
					MOVB	R0, (R11)	
					TSTB	(R10)	1512
					BNEQ	4\$	
					CLRL	-(SP)	1514
					MOVZBL	2(TARGET), -(SP)	

```

M 5
15-Sep-1984 23:57:30      VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:16:44      [DEBUG.SRC]DBGCVTDx.B32;1

```

Page 36  
(6)

PC	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

			17	A2	95	00118	TSTB	23(DESC_VAL)	1573
				0A	12	0011B	BNEQ	15\$	
				50	DD	0011D	PUSHL	DESC_PTR	1576
			08	A2	DD	0011F	PUSHL	8(DESC_VAL)	1575
D8	0000V	CF		02	FB	00122	CALLS	#2, COVER VMSDESC_SETUP	
		53		01	F3	00127	AOBLEQ	#1, I, 12\$	1558
		04		54	D1	0012B	CMPL	FCODE, #4	1586
				05	13	0012E	BEQL	16\$	
		09		54	D1	00130	CMPL	FCODE, #9	
				16	12	00133	BNEQ	17\$	
			08	A8	D5	00135	TSTL	8(R8)	1588
				11	12	00138	BNEQ	17\$	
		02	06	A8	91	0013A	CMPB	6(R8), #2	1591
				0B	12	0013E	BNEQ	17\$	
		6B		6A	90	00140	MOVB	(R10), (R11)	1594
	02	A6	02	A9	90	00143	MOVB	2(TARGET), 2(SOURCE)	1595
		66		69	80	00148	MOVW	(TARGET), (SOURCE)	1596
		14		54	D1	0014B	CMPL	FCODE, #20	1612
				0B	13	0014E	BEQL	18\$	
		08		54	D1	00150	CMPL	FCODE, #8	1616
				49	12	00153	BNEQ	22\$	
		08	06	A8	91	00155	CMPB	6(R8), #8	1622
				09	12	00159	BNEQ	19\$	
18	B7	18	B8	14	A7	28	0015B	18\$: MOV C3	1626
				49	11	00162	BRB	23\$	1622
		52	18	B8	D0	00164	19\$: MOVL	@24(R8), INDEX	1631
				0D	18	00168	BGEQ	20\$	1632
			00028248	8F	DD	0016A	PUSHL	#164424	
	00000000G	00		01	FB	00170	CALLS	#1, LIB\$SIGNAL	
		51	18	A7	D0	00177	20\$: MOVL	24(R7), SETVALUE	1633
		50	14	A7	3C	0017B	MOVZWL	20(R7), R0	1634
		50		0B	C4	0017F	MULL2	#8, R0	
				50	D7	00182	DECL	R0	
		50		52	D1	00184	CMPL	INDEX, R0	
				06	14	00187	BGTR	21\$	
20		61		52	E2	00189	BBSS	INDEX, (SETVALUE), 23\$	1636
				1E	11	0018D	BRB	23\$	
			00028248	8F	DD	0018F	21\$: PUSHL	#164424	1638
	00000000G	00		01	FB	00195	CALLS	#1, LIB\$SIGNAL	
				0F	11	0019C	BRB	23\$	1610
			0C	AC	DD	0019E	22\$: PUSHL	CVT_ROUND_FLAG	1643
			24	AE	9F	001A1	PUSHAB	DUMMY	
			0240	8F	BB	001A4	PUSHR	#*M<R6,R9>	
	0000V	CF		04	FB	001A8	CALLS	#4, DBG\$CVT DX DX	
		6B	10	AE	90	001AD	23\$: MOVB	SOURCE_CLASS, (R11)	1647
	02	A6	08	AE	90	001B1	MOVB	SOURCE_DTYPE, 2(SOURCE)	1648
		66		6E	80	001B6	MOVW	SOURCE_LENGTH, (SOURCE)	1649
		6A	0C	AE	90	001B9	MOVB	TARGET_CLASS, (R10)	1650
	02	A9	04	AE	90	001BD	MOVB	TARGET_DTYPE, 2(TARGET)	1651
		2D	18	AE	E9	001C2	BLBC	24(SP), 24\$	1656
				57	DD	001C6	PUSHL	R7	1660
				7E	D4	001C8	CLRL	-(SP)	1659
				58	DD	001CA	PUSHL	R8	1660
			20	AE	DD	001CC	PUSHL	TYPEID_INDEX	1659
	00000000G	00		04	FB	001CF	CALLS	#4, DBG\$PERFORM_TYPEID_CHECK	
		1C		50	D0	001D6	MOVL	R0, STATUS	
		15	1C	AE	E8	001DA	BLBS	STATUS, 24\$	1661

```

15-Sep-1984 23:57:30      VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:16:44      [DEBUG.SRC]DBGCVTDX.B32;1

```

	00000000G	00	DD	001DE	PUSHL	DBG\$GL_OPCODE_NAME	:	1663
		01	DD	001E4	PUSHL	#1	:	
	0002874B	8F	DD	001E6	PUSHL	#165707	:	
00000000G	00	03	FB	001EC	CALLS	#3, LIB\$SIGNAL	:	
	50	57	DO	001F3	MOVL	R7, R0	:	1666
		04	001F6	RET			:	1667

```
; Routine Size: 503 bytes,    Routine Base: DBG$CODE + 0000
```

```

1541 1668 1 ROUTINE COVER_VMSDESC_SETUP(TYPEID, VMSDESC) =
1542 1669 1
1543 1670 1 FUNCTION
1544 1671 1     This routine is a hack routine called depending on FCODE in the
1545 1672 1     TYPEID. The purpose of this routine is to plunge in the class code
1546 1673 1     and dtype so that the DBG$CVT_DX_DX can be called.
1547 1674 1
1548 1675 1 INPUTS
1549 1676 1     TYPEID - Typeid of the data object.
1550 1677 1
1551 1678 1     VMSDESC - Vax standard Descriptor.
1552 1679 1
1553 1680 1 OUTPUTS
1554 1681 1     VMSDESC is returned.
1555 1682 1
1556 1683 1
1557 1684 2 BEGIN
1558 1685 2 MAP
1559 1686 2     TYPEID: REF RST$ENTRY,
1560 1687 2     VMSDESC: REF BLOCK[,BYTE];
1561 1688 2
1562 1689 2 VMSDESC[DSC$B CLASS] = DSC$K CLASS_S;
1563 1690 2 SELECTONE .TYPEID[RST$B_FCODE] OF
1564 1691 2 SET
1565 1692 2     [RST$K_TYPE_ENUM]:
1566 1693 3 BEGIN
1567 1694 3     SELECTONE .VMSDESC[DSC$W_LENGTH] OF
1568 1695 3     SET
1569 1696 3     [1]:
1570 1697 3         VMSDESC[DSC$B_DTYPE] = DSC$K_DTYPE_BU;
1571 1698 3     [2]:
1572 1699 3         VMSDESC[DSC$B_DTYPE] = DSC$K_DTYPE_WU;
1573 1700 3     [OTHERWISE]:
1574 1701 3         VMSDESC[DSC$B_DTYPE] = DSC$K_DTYPE_LU;
1575 1702 3     TES;
1576 1703 2 END;
1577 1704 2
1578 1705 2 [RST$K_TYPE_SUBRNG]:
1579 1706 3 BEGIN
1580 1707 3 LOCAL
1581 1708 3     DUMMY1, DUMMY2, DUMMY3, DTYPE;
1582 1709 3
1583 1710 3 WHILE .TYPEID[RST$B_FCODE] EQL RST$K_TYPE_SUBRNG DO
1584 1711 3     DBG$STA_TYP_SUBRNG(.TYPEID, TYPEID, DUMMY1, DUMMY2, DUMMY3);
1585 1712 3
1586 1713 3 SELECTONE .TYPEID[RST$B_FCODE] OF
1587 1714 3 SET
1588 1715 3     [RST$K_TYPE_ENUM]:
1589 1716 3         VMSDESC = COVER_VMSDESC_SETUP(.TYPEID, .VMSDESC);
1590 1717 3
1591 1718 3 [RST$K_TYPE_ATOMIC]:
1592 1719 4 BEGIN
1593 1720 4     DBG$STA_TYP_ATOMIC(.TYPEID, DTYPE, DUMMY3);
1594 1721 4     IF .DTYPE EQL DST$K_BOOL THEN DTYPE = DSC$K_DTYPE_TF;
1595 1722 4     VMSDESC[DSC$B_DTYPE] = .DTYPE;
1596 1723 4
1597 1724 4

```

```

: 1598      1725  4
: 1599      1726  4
: 1600      1727  4
: 1601      1728  4
: 1602      1729  4
: 1603      1730  4
: 1604      1731  4
: 1605      1732  5
: 1606      1733  4
: 1607      1734  5
: 1608      1735  5
: 1609      1736  5
: 1610      1737  5
: 1611      1738  5
: 1612      1739  5
: 1613      1740  5
: 1614      1741  5
: 1615      1742  5
: 1616      1743  5
: 1617      1744  4
: 1618      1745  3
: 1619      1746  3
: 1620      1747  3
: 1621      1748  3
: 1622      1749  2
: 1623      1750  2
: 1624      1751  2
: 1625      1752  2
: 1626      1753  2
: 1627      1754  2
: 1628      1755  2
: 1629      1756  2
: 1630      1757  2
: 1631      1758  2
: 1632      1759  2
: 1633      1760  2
: 1634      1761  2
: 1635      1762  1

```

```

: In here we have a bit of problem, for the size of
: the parant and the size of the subrange is different.
: For example, subrange's parant can be longword integer
: and subrange can be 1 byte. In order for the type
: converter to take the right value, we adjust the dtype
: by the length.
IF (.DTYPE NEQ DSC$K_DTYPE_T AND .DTYPE NEQ DSC$K_DTYPE_TF)
THEN
  BEGIN
    SELECTONE .VMSDESC[DSC$W_LENGTH] OF
      SET
      [1]: VMSDESC[DSC$B_DTYPE] = DSC$K_DTYPE_B;
      [2]: VMSDESC[DSC$B_DTYPE] = DSC$K_DTYPE_W;
      [OTHERWISE]:
        VMSDESC[DSC$B_DTYPE] = DSC$K_DTYPE_L;
      TES;
    END;
  END;
  TES;
END;
[ RST$K_TYPE SET, RST$K_TYPE TPTR]:
  VMSDESC[DSC$B_DTYPE] = DSC$K_DTYPE_L;
[ RST$K_TYPE RFA]:
  VMSDESC[DSC$B_CLASS] = 0;
[ OTHERWISE]:
  $DBG_ERROR('DBGCVTDX\COVER_VMSDESC_SETUP');
  TES;
RETURN .VMSDESC;
END;

```

.PSECT DBG\$PLIT,NOWRT, SHR, PIC,0

```

52 45 56 4F 43 5C 58 44 54 56 43 47 42 44 1C 00AB5 P.ADW: .ASCII <28>\DBGCVTDX\<92>\COVER_VMSDESC_SETUP\
50 55 54 45 53 5F 43 53 45 44 53 4D 56 5F 00AC4

```

.PSECT DBG\$CODE,NOWRT, SHR, PIC,0

```

000C 00000 COVER_VMSDESC_SETUP:
      5E      10 C2 00002      .WORD Save R2,R3      : 1668
      52      08 AC D0 00005      SUBL2 #16, SP      : 1689
03    A2      01 90 00009      MOVL VMSDESC, R2
      50      04 AC D0 0000D      MOVB #1, 3(R2)
      50      18 C0 00011      MOVL TYPEID, R0      : 1690
      ADDL2 #24, R0

```

	50		60	9A	00014	MOVZBL	(R0), R0		
	04		50	91	00017	CMPB	R0, #4	1692	
			1C	12	0001A	BNEQ	3\$		
	01		62	B1	0001C	CMPW	(R2), #1	1696	
			06	12	0001F	BNEQ	1\$		
02	A2		02	90	00021	MOVB	#2, 2(R2)	1697	
			53	11	00025	BRB	6\$		
	02		62	B1	00027	CMPW	(R2), #2	1698	
			06	12	0002A	BNEQ	2\$		
02	A2		03	90	0002C	MOVB	#3, 2(R2)	1699	
			48	11	00030	BRB	6\$		
02	A2		04	90	00032	MOVB	#4, 2(R2)	1701	
			42	11	00036	BRB	6\$	1690	
	09		50	91	00038	CMPB	R0, #9	1705	
			03	13	0003B	BEQL	4\$		
		0085	31	0003D	BRW	10\$			
	50	04	AC	D0	00040	MOVL	TYPEID, R0	1710	
	09	18	A0	91	00044	CMPB	24(R0), #9		
			17	12	00048	BNEQ	5\$		
		08	AE	9F	0004A	PUSHAB	DUMMY3	1711	
		04	AE	9F	0004D	PUSHAB	DUMMY2		
		0C	AE	9F	00050	PUSHAB	DUMMY1		
		04	AC	9F	00053	PUSHAB	TYPEID		
			50	DD	00056	PUSHL	R0		
00000000G	00		05	FB	00058	CALLS	#5, DBG\$STA_TYP_SUBRNG		
			DF	11	0005F	BRB	4\$		
	53	04	AC	D0	00061	MOVL	TYPEID, R3	1713	
	50	18	A3	9A	00065	MOVZBL	24(R3), R0		
	04		50	91	00069	CMPB	R0, #4	1715	
			0E	12	0006C	BNEQ	7\$		
			52	DD	0006E	PUSHL	R2	1716	
			53	DD	00070	PUSHL	R3		
8A	AF		02	FB	00072	CALLS	#2, COVER_VMSDESC_SETUP		
08	AC		50	D0	00076	MOVL	R0, VMSDESC		
			78	11	0007A	BRB	14\$		
	02		50	91	0007C	CMPB	R0, #2	1718	
			73	12	0007F	BNEQ	14\$		
		08	AE	9F	00081	PUSHAB	DUMMY3	1720	
		10	AE	9F	00084	PUSHAB	DTYPE		
			53	DD	00087	PUSHL	R3		
00000000G	00		03	FB	00089	CALLS	#3, DBG\$STA_TYP_ATOMIC		
00000009E	8F	0C	AE	D1	00090	CMPL	DTYPE, #158	1721	
			04	12	00098	BNEQ	8\$		
0C	AE		28	D0	0009A	MOVL	#40, DTYPE		
02	A2	0C	AE	90	0009E	MOVB	DTYPE, 2(R2)	1722	
	0E	0C	AE	D1	000A3	CMPL	DTYPE, #14	1732	
			4B	13	000A7	BEQL	14\$		
	28	0C	AE	D1	000A9	CMPL	DTYPE, #40		
			45	13	000AD	BEQL	14\$		
	01		62	B1	000AF	CMPW	(R2), #1	1737	
			06	12	000B2	BNEQ	9\$		
02	A2		06	90	000B4	MOVB	#6, 2(R2)	1738	
			3A	11	000B8	BRB	14\$		
	02		62	B1	000BA	CMPW	(R2), #2	1739	
			10	12	000BD	BNEQ	11\$		
02	A2		07	90	000BF	MOVB	#7, 2(R2)	1740	
			2F	11	000C3	BRB	14\$		

DBGCVTDX  
V04-000

N 5  
15-Sep-1984 23:57:30 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:16:44 [DEBUG.SRC]DBGCVTDX.B32;1

Page 42  
(7)

06	50	91	000C5	10\$:	CMPB	R0	#6	:	1751
	05	13	000C8		BEQL	11\$		:	
08	50	91	000CA		CMPB	R0	#8	:	
	06	12	000CD		BNEQ	12\$		:	
02	A2	08	90 000CF	11\$:	MOVB	#8	2(R2)	:	1752
	1F	11	000D3		BRB	14\$		:	
14	50	91	000D5	12\$:	CMPB	R0	#20	:	1754
	05	12	000D8		BNEQ	13\$		:	
	03	A2	94 000DA		CLRB	3(R2)		:	1755
	15	11	000DD		BRB	14\$		:	
		EF	9F 000DF	13\$:	PUSHAB	P.ADW		:	1758
		01	DD 000E5		PUSHL	#1		:	
		8F	DD 000E7		PUSHL	#164706		:	
00000000G	00	03	FB 000ED		CALLS	#3, LIB\$SIGNAL		:	
	50	08	AC D0 000F4	14\$:	MOVL	VMSDESC, R0		:	1761
		04	000F8		RET			:	1762

; Routine Size: 249 bytes, Routine Base: DBG\$CODE + 01F7



```

: 1637 1763 1 GLOBAL ROUTINE DBG$CVT_DX_DX (SOURCE, DESTINATION, OUTLEN): NOVALUE =
: 1638 1764 1
: 1639 1765 1 This is the general data type conversion facility.
: 1640 1766 1 Given two parameters, one the source descriptor,
: 1641 1767 1 second the destination descriptor this routine
: 1642 1768 1 will convert the source to destination.
: 1643 1769 1 The permitted set of class, data type and combinat on
: 1644 1770 1 of the two is a subset of the ones allowed in the
: 1645 1771 1 calling standard.
: 1646 1772 1
: 1647 1773 1 The following is a general description of DBG$CVT_DX_DX.
: 1648 1774 1
: 1649 1775 1 This module is divided into two routines on the bases of functional
: 1650 1776 1 modularity. The front-end (FIND_CVT_PATH), and back-end (DBG$CVT_DX_DX).
: 1651 1777 1 The front-end converts the source into an intermediate data type, and
: 1652 1778 1 frees the back-end of any error checking of invalid classes and/or
: 1653 1779 1 data types (or combination of the two), and of decisions that require
: 1654 1780 1 knowledge of which class or data type is being converted. The only
: 1655 1781 1 information that the back-end knows about is what the conversion path
: 1656 1782 1 is, and where the intermediate data is. The back-end then scales the
: 1657 1783 1 intermediate data if necessary and converts it to the destination
: 1658 1784 1 data type. Note that even though a scale may not be necessary, the
: 1659 1785 1 intermediate data is still converted to a second intermediate data type
: 1660 1786 1 just to be consistent.
: 1661 1787 1
: 1662 1788 1 1. Upon entry to DBG$CVT_DX_DX, FIND_CVT_PATH routine is called.
: 1663 1789 1 FIND_CVT_PATH has 4 functions, they are:
: 1664 1790 1 a. Find any errors concerning the class and data type of
: 1665 1791 1 source and destination descriptor. These errors can be
: 1666 1792 1 invalid class, invalid data type, or invalid combination
: 1667 1793 1 of a class and data type. It can also tell which descriptors
: 1668 1794 1 are supported by the VAX-11 calling standard and which are
: 1669 1795 1 supported by this routine.
: 1670 1796 1
: 1671 1797 1 b. Figure out what the conversion path is, i.e. class,dtype --> class,dtype.
: 1672 1798 1 These paths are given names such as K_SMLINT_DEC, which reads
: 1673 1799 1 "from small integer to decimal" (categories are defined later).
: 1674 1800 1
: 1675 1801 1 c. Convert the source data to an intermediate data. The strategy
: 1676 1802 1 used to select the appropriate intermediate data is explained later.
: 1677 1803 1 Precision should not be lost in converting to the intermediate type.
: 1678 1804 1
: 1679 1805 1 d. Put whatever information needed about the source and destination
: 1680 1806 1 descriptor in two structures passed by DBG$CVT_DX_DX.
: 1681 1807 1 These two structures SRC_INFO, and DST_INFO, contain the kind
: 1682 1808 1 of information that can only be visible when the class, and
: 1683 1809 1 data type of the source and destination descriptors are being
: 1684 1810 1 manipulated. These two structures can be expanded to contain
: 1685 1811 1 more information as new class, and data types may require it.
: 1686 1812 1
: 1687 1813 1
: 1688 1814 1 2. The following is an overview of the design of FIND_CVT_PATH:
: 1689 1815 1 The problem to be solved is to recognize "valid" descriptors.
: 1690 1816 1 A descriptor is valid if the CLASS and DATA TYPE fields of the
: 1691 1817 1 descriptor satisfy certain conditions.
: 1692 1818 1 With this problem in mind we shall use some formal language theory
: 1693 1819 1 and applications to solve it.

```

```

1694 1820 1
1695 1821 1
1696 1822 1
1697 1823 1
1698 1824 1
1699 1825 1
1700 1826 1
1701 1827 1
1702 1828 1
1703 1829 1
1704 1830 1
1705 1831 1
1706 1832 1
1707 1833 1
1708 1834 1
1709 1835 1
1710 1836 1
1711 1837 1
1712 1838 1
1713 1839 1
1714 1840 1
1715 1841 1
1716 1842 1
1717 1843 1
1718 1844 1
1719 1845 1
1720 1846 1
1721 1847 1
1722 1848 1
1723 1849 1
1724 1850 1
1725 1851 1
1726 1852 1
1727 1853 1
1728 1854 1
1729 1855 1
1730 1856 1
1731 1857 1
1732 1858 1
1733 1859 1
1734 1860 1
1735 1861 1
1736 1862 1
1737 1863 1
1738 1864 1
1739 1865 1
1740 1866 1
1741 1867 1
1742 1868 1
1743 1869 1
1744 1870 1
1745 1871 1
1746 1872 1
1747 1873 1
1748 1874 1
1749 1875 1
1750 1876 1

```

Let us take a hypothetical problem that is very close but smaller in magnitude of the original problem and solve it.  
Suppose that the set of classes that we are interested in are  
CLASS = { c1, c2, c3 }, and the set of data types are  
DTYPE = { d1, d2, d3, d4 }. Then suppose that only a certain combinations of CLASS and DTYPE are valid, and they are c1d3, c2d1, c3d2, c3d4.  
Hence language L(G) is consisted of sentences { c1d3, c2d1, c3d2, c3d4 }.  
First we need to come up with a grammar for the language L(G).

Grammar for L(G) :

```

Z --> <S1>d3 ; <S2>d1 ; <S3>d2 ; <S3>d4
S1 --> c1
S2 --> c2
S3 --> c3
S4 --> c4

```

A close look shows that this is a Chomsky type 3 regular grammar, because productions are all

NON-TERMINAL --> terminal

or

NON-TERMINAL --> <NON-TERMINAL>terminal

This type of grammar has the nice feature that its sentential forms can be "accepted" by a finite state machine.

The sentential forms of this grammar can also be accepted by a deterministic finite automaton (DFA) because each right hand side has a unique left hand side.

A DFA can be written to recognize sentences of this grammar and to reject sentences that are not in the language.

The original problem is very similar to this hypothetical one, the only difference is that the set of CLASSES and DTYPES is larger.

FIND\_CVT\_PATH is just a DFA that accepts sentences of language L(V) when L(V) is pairs of VAX-11 DSC\$K\_CLASS x DSC\$K\_DTYPE y. The grammar for L(V) is very similar to the grammar for L(G) above.

3. In order to achieve the conflicting goals: fast, not large in size, expandable, no loss of precision as a result of intermediate values, there is a need for a compromise. The strategy for categorizing the data types is based on three goals: precision should not be lost as a result of converting to intermediate data types, data types of the same category should share similar internal representations so they can be converted to and from each other easily, and data types that have to be converted through software should be separated from those that have associated machine instructions. The third goal provides easy and fast conversions for those data types with associated machine instructions.
- The current categories were formulated by the following strategy:
- Divide the integers into two groups, small and large integers.
  - Divide the floating numbers into two categories small and large floating. The small category will be the data types that machine instructions are available for their conversions.
  - The large category consist of data types that there are no machine instructions for their conversions or the instructions must be emulated (LIB\$EMULATE) for some VAX machines.

This categorization will provide conversions that are fast and smooth. As a result we have the following :

```

INTEGER --> SMALL_INTEGER ; LARGE_INTEGER
FLOAT --> SMALL_FLOAT ; LARGE_FLOAT
SMALL_INTEGER --> bu ; wu ; b ; w ; l !Intermediate L

```

```
1751 1877 1 : LARGE_INTEGER --> lu : q !Intermediate OU
1752 1878 1 : SMALL_FLOAT --> f : d !Intermediate D
1753 1879 1 : LARGE_FLOAT --> g : h !Intermediate H
1754 1880 1 : DEC --> nu : nl : nlo : nr : nro : nz !Intermediate P
1755 1881 1 : NBDS --> nbds !Intermediate T
1756 1882 1 :
1757 1883 1 :
1758 1884 1 : 4. Upon return from FIND_CVT_PATH, the main routine then enters a
1759 1885 1 : CASE statement that selects the desired conversion. This CASE
1760 1886 1 : is explained in detail in the first paragraph of the statement.
1761 1887 1 :
1762 1888 1 : AUTHOR: Farokh Morshed 01-09-1981
1763 1889 1 :
1764 1890 1 : FUNCTIONAL DESCRIPTION:
1765 1891 1 : Upon entry, FIND_CVT_PATH is called to identify which conversion is to be
1766 1892 1 : done, i.e. from which CLASS, DTYPE combination to which CLASS, DTYPE
1767 1893 1 : combination.
1768 1894 1 : Also, FIND_CVT_PATH will do all the work of identifying the errors such
1769 1895 1 : as unsupported class, data type, or combinations.
1770 1896 1 : This routine is just a tree of CASE statements, where the outermost
1771 1897 1 : level CASE statement labels have been determined by the FIND_CVT_PATH
1772 1898 1 : routine.
1773 1899 1 :
1774 1900 1 :
1775 1901 1 : CALLING SEQUENCE:
1776 1902 1 :
1777 1903 1 : ret_status.wlc.v = DBG$CVT_DX_DX ( SOURCE.rx.dx, DESTINATION.wx.dx
1778 1904 1 : <OUTLEN.wwu.r> <CVT_ROUND_FLAG>)
1779 1905 1 :
1780 1906 1 : FORMAL PARAMETERS:
1781 1907 1 :
1782 1908 1 : SOURCE Address of source descriptor.
1783 1909 1 : DESTINATION Address of destination descriptor.
1784 1910 1 : OUTLEN Output length. Optional parameter for this
1785 1911 1 : routine to put the length of actual data (without padding) in.
1786 1912 1 : This is used only when destination is of data
1787 1913 1 : type T.
1788 1914 1 : CVT_ROUND_FLAG A flag set to true to indicate the conversion
1789 1915 1 : result is rounded.
1790 1916 1 :
1791 1917 1 : IMPLICIT INPUTS:
1792 1918 1 :
1793 1919 1 : NONE
1794 1920 1 :
1795 1921 1 : IMPLICIT OUTPUTS:
1796 1922 1 :
1797 1923 1 : NONE
1798 1924 1 :
1799 1925 1 : SIDE EFFECTS
1800 1926 1 : Every routine in this module turns on every arithmetic trap in PSW.
1801 1927 1 : Caller must have LIB$EMULATE as handler if any G or H conversions are
1802 1928 1 : asked for.
1803 1929 1 :
1804 1930 2 : BEGIN
1805 1931 2 :
1806 1932 2 : LOCAL
1807 1933 2 : CVT_ROUND_FLAG,
```

1808	1934	2	SRC_INFO: BLOCK [K_SRC_INFO_LENGTH, BYTE] FIELD (SRC_INFO_FIELDS),	! Source information structu
1809	1935	2	DST_INFO: BLOCK [K_DST_INFO_LENGTH, BYTE] FIELD (DST_INFO_FIELDS),	! Destination information st
1810	1936	2	OUTPUT_BUFFER: BLOCK [K_OUTPUT_BUFFER_LENGTH, BYTE],	
1811	1937	2	OUTPUT,	
1812	1938	2	DESTINATION_PTR,	! Ptr to destination
1813	1939	2	INTERMED_DATA: BLOCK [K_INTERMED_DATA_LENGTH, BYTE],	! Intermediate data buffer.
1814	1940	2	TEMP_BUF1: BLOCK [K_TEMP_BUF_LENGTH, BYTE],	! Holds temporary data.
1815	1941	2	TEMP_BUF2: BLOCK [K_TEMP_BUF_LENGTH, BYTE],	! Holds temporary data.
1816	1942	2	CLASS_S_DESC: BLOCK [8, BYTE],	! A class S descriptor for a
1817	1943	2	CLASS_SOURCE: BYTE,	! Class of source VMS descri
1818	1944	2	CLASS_TARGET: BYTE,	! Class of target VMS descri
1819	1945	2	FINAL_LEN,	! Length of data in TEMP_BUF
1820	1946	2	CVT_PATH,	! Calculated convert path.
1821	1947	2	NO_DIGITS,	! Number of digits in decima
1822	1948	2	DIGITS_IN_FRACT,	! Fractional digits (OTSSCVT
1823	1949	2	FLOAT_SCALE,	! Scale of a floating number
1824	1950	2	SIGN,	! Sign of the floating numbe
1825	1951	2	STATUS,	! Return status.
1826	1952	2	LRGST_P_LU,	! Largest LU in a packed dec
1827	1953	2	LRGST_D_LU,	! Largest LU in a double flo
1828	1954	2	PACK_ZERO,	! A zero in a packed decimal
1829	1955	2	LRGST_H_LU,	! Largest LU in a H floating
1830	1956	2	BUF_OFFSET,	! Offset to actual data in f
1831	1957	2	NEXT_BLANK,	! Location of next blank in
1832	1958	2	OUTPUT_STR_LEN,	! Length of actual string th
1833	1959	2	SRC_POS,	! Bit offset in source.
1834	1960	2	DST_POS,	! Bit offset in destination.
1835	1961	2		! to destination (optional p
1836	1962	2	BIN_SCALE,	
1837	1963	2	SCALE;	! Effective scale (source sc
1838	1964	2		
1839	1965	2	MAP	
1840	1966	2	OUTPUT: REF BLOCK[, BYTE]	
1841	1967	2	DESTINATION_PTR: REF BLOCK[, BYTE],	
1842	1968	2	SOURCE: REF BLOCK[, BYTE],	! Source VMS descriptor.
1843	1969	2	DESTINATION: REF BLOCK[, BYTE];	! Destination VMS descriptor
1844	1970	2		
1845	1971	2	BUILTIN	
1846	1972	2	ACTUALPARAMETER,	
1847	1973	2	ACTUALCOUNT;	
1848	1974	2		
1849	1975	2		
1850	1976	2	! Establish CVT_HANDLER as handler.	
1851	1977	2	!	
1852	1978	2	ENABLE	
1853	1979	2	CVT_HANDLER;	
1854	1980	2		
1855	1981	2		
1856	1982	2	! These literals are used a few lines down to test whether we	
1857	1983	2	! are doing a conversion from decimal string or packed.	
1858	1984	2	!	
1859	1985	2	LITERAL	
1860	1986	2	MIN_DEC_DTYPE = DSC\$K_DTYPE_NU, ! 15	
1861	1987	2	MAX_DEC_DTYPE = DSC\$K_DTYPE_P; ! 21	
1862	1988	2		
1863	1989	2	!++	
1864	1990	2	! If the destination or source is Absolute Date Time cut it off here	

1865	1991	2	! and do the conversion.	
1866	1992	2	! _	
1867	1993	2	IF (.DESTINATION [DSC\$B_DTYPE] EQL DSC\$K_DTYPE_ADT) OR	
1868	1994	2	(.SOURCE [DSC\$B_DTYPE] EQL DSC\$K_DTYPE_ADT)	
1869	1995	2	THEN	
1870	1996	2	IF (.DESTINATION [DSC\$B_DTYPE] EQL DSC\$K_DTYPE_T) AND	ADT to text
1871	1997	2	(.SOURCE [DSC\$B_DTYPE] EQL DSC\$K_DTYPE_ADT)	
1872	1998	2	THEN	
1873	1999	2	BEGIN	
1874	2000	2	LOCAL	
1875	2001	2	TEMP;	
1876	2002	2	CLASS_S_DESC[dsc\$b_class] = dsc\$k_class_s;	Build the string descripto
1877	2003	2	CLASS_S_DESC[dsc\$b_dtype] = dsc\$k_dtype_t;	
1878	2004	2	CLASS_S_DESC[dsc\$w_length] = 23;	
1879	2005	2	CLASS_S_DESC[dsc\$a_pointer] = .DESTINATION [DSC\$A_POINTER];	
1880	2006	2	IF NOT (SYSSASCTIM (TEMP, CLASS_S_DESC, .SOURCE [DSC\$A_POINTER], 0))	
1881	2007	2	THEN	
1882	2008	2	SIGNAL( DBG\$DELTIMTOO );	
1883	2009	2	OUTPUT_STR_LEN = .TEMP;	
1884	2010	2	END	
1885	2011	2	ELSE	
1886	2012	2	IF (.DESTINATION [DSC\$B_DTYPE] EQL DSC\$K_DTYPE_ADT) AND	Text to ADT
1887	2013	2	(.SOURCE [DSC\$B_DTYPE] EQL DSC\$K_DTYPE_T)	
1888	2014	2	THEN	
1889	2015	2	BEGIN	
1890	2016	2	CLASS_S_DESC[dsc\$b_class] = dsc\$k_class_s;	Build the string descripto
1891	2017	2	CLASS_S_DESC[dsc\$b_dtype] = dsc\$k_dtype_t;	
1892	2018	2	CLASS_S_DESC [DSC\$W_LENGTH] = .SOURCE [DSC\$W_LENGTH];	
1893	2019	2	CLASS_S_DESC [DSC\$A_POINTER] = .SOURCE [DSC\$A_POINTER];	
1894	2020	2	IF NOT (SYSSBINTIM (CLASS_S_DESC, .DESTINATION [DSC\$A_POINTER]))	
1895	2021	2	THEN	
1896	2022	2	SIGNAL( DBG\$_ABSDATSYN );	He didn't like to text for
1897	2023	2	END	
1898	2024	2	ELSE	
1899	2025	2	SIGNAL( DBG\$_ILLTYPE )	Nothing but ADT to text or
1900	2026	2	ELSE	
1901	2027	2	BEGIN	
1902	2028	2	! Get the conversion rounding flag. TRUE = round.	
1903	2029	2	IF ACTUALCOUNT() GTR 3	
1904	2030	2	THEN	
1905	2031	2	CVT_ROUND_FLAG = ACTUALPARAMETER(4)	
1906	2032	2	ELSE	
1907	2033	2	CVT_ROUND_FLAG = FALSE;	
1908	2034	2		
1909	2035	2		
1910	2036	2		
1911	2037	2	! Strip the non-significant zeros for packed decimal.	
1912	2038	2	IF .SOURCE[DSC\$B_DTYPE] EQL DSC\$K_DTYPE_P AND	
1913	2039	2	.DESTINATION[DSC\$B_DTYPE] NEQ DSC\$K_DTYPE_T	
1914	2040	2	THEN	
1915	2041	2	SOURCE = DBG\$STRIP_ZEROES(.SOURCE);	
1916	2042	2		
1917	2043	2		
1918	2044	2		
1919	2045	2	! This flag is so that a special error can	
1920	2046	2	! be signalled for reserved operand during a decimal string conversion.	
1921	2047	2	! Note that this test relies on the fact that the dtypes for the	

```

1922 2048 3 | decimal string data types are cover the range from the MIN_DEC_DTYPE
1923 2049 3 | code to the MAX_DEC_DTYPE code.
1924 2050 3 |
1925 2051 3 | DECIMAL CONVERT =
1926 2052 3 |   (.SOURCE[DSC$B_DTYPE] GEQ MIN_DEC_DTYPE) AND
1927 2053 3 |   (.SOURCE[DSC$B_DTYPE] LEQ MAX_DEC_DTYPE);
1928 2054 3 |
1929 2055 3 |
1930 2056 3 | DESTINATION_PTR is used to indicate the destination
1931 2057 3 | of the converted data. If the data type is unaligned, then
1932 2058 3 | the output-buffer pointer points to a temporary buffer. Else,
1933 2059 3 | the output-buffer pointer points to the caller's buffer.
1934 2060 3 |
1935 2061 3 | IF .DESTINATION[DSC$B_CLASS] EQL DSC$K_CLASS_UBS
1936 2062 3 | THEN
1937 2063 4 |   BEGIN
1938 2064 4 |     OUTPUT = OUTPUT_BUFFER;
1939 2065 4 |     DESTINATION_PTR = .DESTINATION[DSC$A_POINTER];
1940 2066 4 |   END
1941 2067 3 | ELSE
1942 2068 3 |   OUTPUT = .DESTINATION[DSC$A_POINTER];
1943 2069 3 |
1944 2070 3 |
1945 2071 3 | Zero and blank out these records for FIND_CVT_PATH.
1946 2072 3 |
1947 2073 3 | CH$FILL (0, K_SRC_INFO_LENGTH, SRC_INFO);
1948 2074 3 | CH$FILL (0, K_DST_INFO_LENGTH, DST_INFO);
1949 2075 3 | CH$FILL (0, K_INTMED_DATA_LENGTH, INTMED_DATA);
1950 2076 3 | CH$FILL (%C' ', K_TEMP_BUF_LENGTH, TEMP_BUF1);
1951 2077 3 | CH$FILL (%C' ', K_TEMP_BUF_LENGTH, TEMP_BUF2);
1952 2078 3 | OUTPUT_STR_LEN = 0;
1953 2079 3 |
1954 2080 3 |
1955 2081 3 | This descriptor is always class S, dtype T.
1956 2082 3 | It is used on various occasions to call routines that require
1957 2083 3 | descriptors as their parameters.
1958 2084 3 |
1959 2085 3 | CLASS_S_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
1960 2086 3 | CLASS_S_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
1961 2087 3 |
1962 2088 3 |
1963 2089 3 | Initialize some constants.
1964 2090 3 |
1965 2091 3 | LRGST_P_LU = UPLIT (%P'+4294967295');
1966 2092 3 | LRGST_D_LU = UPLIT (%D'+4294967295');
1967 2093 3 | LRGST_H_LU = UPLIT (%H'+4294967295');
1968 2094 3 | PACK_ZERO = UPLIT (%P'+0');
1969 2095 3 |
1970 2096 3 |
1971 2097 3 | SRC_INFO structure will contain the information about the source data. In
1972 2098 3 | most cases it will point to the INTMED_DATA buffer because the source data is
1973 2099 3 | usually converted to an intermediate, so before calling FIND_CVT_PATH we
1974 2100 3 | set up the pointer and length fields of SRC_INFO to be INTMED_DATA.
1975 2101 3 |
1976 2102 3 | SRC_INFO [S_POINTER] = INTMED_DATA;
1977 2103 3 | SRC_INFO [S_LEN] = K_INTMED_DATA_LENGTH;
1978 2104 3 |

```

1979 2105  
1980 2106  
1981 2107  
1982 2108  
1983 2109  
1984 2110  
1985 2111  
1986 2112  
1987 2113  
1988 2114  
1989 2115  
1990 2116  
1991 2117  
1992 2118  
1993 2119  
1994 2120  
1995 2121  
1996 2122  
1997 2123  
1998 2124  
1999 2125  
2000 2126  
2001 2127  
2002 2128  
2003 2129  
2004 2130  
2005 2131  
2006 2132  
2007 2133  
2008 2134  
2009 2135  
2010 2136  
2011 2137  
2012 2138  
2013 2139  
2014 2140  
2015 2141  
2016 2142  
2017 2143  
2018 2144  
2019 2145  
2020 2146  
2021 2147  
2022 2148  
2023 2149  
2024 2150  
2025 2151  
2026 2152  
2027 2153  
2028 2154  
2029 2155  
2030 2156  
2031 2157  
2032 2158  
2033 2159  
2034 2160  
2035 2161

```

: Call FIND_CVT_PATH to get information on the source and destination
: (SRC_INFO and DST_INFO), and to determine the conversion path
: (CVT_PATH).
STATUS = FIND_CVT_PATH (.SOURCE, .DESTINATION, SRC_INFO, DST_INFO, CVT_PATH);

: If we got an error returned to us by FIND_CVT_PATH, it means that one of the
: descriptors - SOURCE or DESTINATION - was invalid to this routine.
: Errors are represented as negative values. They are listed in the completion
: status section of FIND_CVT_PATH. Although we get a variety of errors,
: from -1 to -7, overlapping can occur.
IF .STATUS LSS 0
THEN
  BEGIN
    CASE .STATUS FROM K_INVNBDS TO K_UNSCAROU OF
      SET
        [K_UNSDTYSTA, K_UNSDTYROU]: $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: invalid dtype in descriptor')
        [K_UNSCLASTA, K_UNSCAROU]: $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: invalid class in descriptor')
        [K_UNSDESSTA, K_UNSDESROU]: $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: invalid class-dtype combinati
        [K_INVNBDS]: $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: invalid numeric byte string d
      TES;
    END;

: Enable all arithmetic traps, and figure out the scale fator to be used by
: the main CASE statement below. The scale factor in SCALE will be a decimal
: scale factor. The scale factor in BIN_SCALE will be a binary scale factor.
BISPSW (%REF (K SET ARITHMETIC TRAP));
SCALE = (IF .SRC_INFO[S_BIN_SCALE] THEN 0 ELSE .SRC_INFO [S_SCALE]) -
        (IF .DST_INFO[D_BIN_SCALE] THEN 0 ELSE .DST_INFO [D_SCALE]);
BIN_SCALE = (IF .SRC_INFO[S_BIN_SCALE] THEN .SRC_INFO [S_SCALE] ELSE 0) -
            (IF .DST_INFO[D_BIN_SCALE] THEN .DST_INFO [D_SCALE] ELSE 0);

: We now have SRC_INFO, DST_INFO, and CVT_PATH, and the source data
: has been converted to an intermediate type. Next step: to go from
: the intermediate form to a scaled version to the actual data type
: called for by the destination descriptor.

: The following explains the objective of the conversions:

: The objective is to convert from intermediate data type provided by
: FIND_CVT_PATH routine to the data type that the user has requested in
: the destination descriptor.

: The intermediate data is in INTMED_DATA, except for when source is
: of data type I. FIND_CVT_PATH does not convert or transform the I
: data types, so the intermediate data for this data type is described
: by the SOURCE descriptor itself.

: The first step is to scale the intermediate data. The scale is
: calculated as: SCALE = (source scale) - (destination scale).
: Scaling cannot always be done on the intermediate data because there

```

```

: 2036 2162 3
: 2037 2163 3
: 2038 2164 3
: 2039 2165 3
: 2040 2166 3
: 2041 2167 3
: 2042 2168 3
: 2043 2169 3
: 2044 2170 3
: 2045 2171 3
: 2046 2172 3
: 2047 2173 3
: 2048 2174 3
: 2049 2175 3
: 2050 2176 3
: 2051 2177 3
: 2052 2178 3
: 2053 2179 3
: 2054 2180 3
: 2055 2181 3
: 2056 2182 3
: 2057 2183 3
: 2058 2184 3
: 2059 2185 3
: 2060 2186 3
: 2061 2187 3
: 2062 2188 3

```

may be under/over flow, so scaling is done on either the intermediate or the highest data type of the category that the destination data type falls in. The data type with greater range is always selected. Caution is taken not to select a scaling intermediate data type that requires G, H, or O instructions, unless source or destination is of these types.

At the beginning of each sub-case statement, there is a macro; each macro is type specific, and scales the intermediate data type involved in that sub-case.

Regardless of whether there is scaling involved or not the intermediate data type is converted to scaling intermediate data type. The scaled intermediate data will again end up in INTMED\_DATA buffer.

Macros that do this scaling are called M\_SCALE\_x\_y: convert x to y, where the result value in y is scaled according to the scale specified in source and destination descriptors.

The next step is to convert the scaled intermediate data to destination data type and move it to where the destination address points to. This is done as close to a 'interrupt proof' manner as possible. Since only NBDS can be of semantics other than fixed, only in case of NBDS (or just text) is the destination copied via a RTL call (LIB\$SCOPY\_x).

PSW is masked such that IV, FU, DV bits are set.

CASE .CVT\_PATH FROM K\_SMLINT\_SMLINT TO K\_NBDS\_NBDS OF  
SET



```

: 2064 2189 3
: 2065 2190 3
: 2066 2191 4
: 2067 2192 4
: 2068 2193 4
: 2069 2194 4
: 2070 2195 4
: 2071 2196 4
: 2072 2197 5
: 2073 2198 5
: 2074 2199 5
: 2075 2200 4
: 2076 2201 4
: 2077 2202 4
: 2078 2203 5
: 2079 2204 5
: 2080 2205 5
: 2081 2206 4
: 2082 2207 4
: 2083 2208 4
: 2084 2209 4
: 2085 2210 4
: 2086 2211 4
: 2087 2212 4
: 2088 2213 4
: 2089 2214 4
: 2090 2215 4
: 2091 2216 4
: 2092 2217 4
: 2093 2218 5
: 2094 2219 5
: 2095 2220 5
: 2096 2221 5
: 2097 2222 5
: 2098 2223 5
: 2099 2224 6
: 2100 2225 6
: 2101 2226 5
: 2102 2227 4
: 2103 2228 4
: 2104 2229 4
: 2105 2230 4
: 2106 2231 4
: 2107 2232 3

```

```

[K_SMLINT_SMLINT]:
BEGIN
M_SCALE L L;
CASE .DESTINATION [DSC$B_DTYPE] FROM DSC$K_DTYPE_V TO DSC$K_DTYPE_SVU OF
SET

[DSC$K_DTYPE_BU]:
BEGIN
IF (OUTPUT [BYTE 1] = .INTMED_DATA [LONG 1]) GTRU K_LRGST_BU
THEN SIGNAL (DBG$_IINTOVF, 1, .DBG$GL_OPCODE_NAME);
END;

[DSC$K_DTYPE_WU]:
BEGIN
IF (OUTPUT [WORD 1] = .INTMED_DATA [S_LONG 1]) GTRU K_LRGST_WU
THEN SIGNAL (DBG$_IINTOVF, 1, .DBG$GL_OPCODE_NAME);
END;

[DSC$K_DTYPE_B]:
CVTLB (INTMED_DATA, .OUTPUT);

[DSC$K_DTYPE_W]:
CVTLW (INTMED_DATA, .OUTPUT);

[DSC$K_DTYPE_L]:
OUTPUT [LONG 1] = .INTMED_DATA [S_LONG 1];

[DSC$K_DTYPE_V, DSC$K_DTYPE_SV, DSC$K_DTYPE_VU, DSC$K_DTYPE_SVU, DSC$K_DTYPE_TF]:
BEGIN
MAP
OUTPUT: REF BITVECTOR[K_OUTPUT_BUFFER_LENGTH * 8],
INTMED_DATA: BITVECTOR[R_INTMED_DATA_LENGTH * 8];

INCR I FROM 0 TO .DST_INFO[D_LEN]-1 DO
BEGIN
OUTPUT[I] = .INTMED_DATA[I];
END;
END;

[INRANGE, OUTRANGE]:
$DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: smlint smlint');
TES;
!For SMLINT_SMLINT
END;

```

2109	2233	3
2110	2234	3
2111	2235	4
2112	2236	4
2113	2237	4
2114	2238	4
2115	2239	4
2116	2240	5
2117	2241	5
2118	2242	4
2119	2243	4
2120	2244	4
2121	2245	5
2122	2246	5
2123	2247	4
2124	2248	4
2125	2249	4
2126	2250	4
2127	2251	4
2128	2252	4
2129	2253	4
2130	2254	5
2131	2255	5
2132	2256	5
2133	2257	5
2134	2258	5
2135	2259	5
2136	2260	5
2137	2261	4
2138	2262	4
2139	2263	4
2140	2264	5
2141	2265	5
2142	2266	5
2143	2267	5
2144	2268	5
2145	2269	5
2146	2270	5
2147	2271	6
2148	2272	6
2149	2273	6
2150	2274	7
2151	2275	7
2152	2276	7
2153	2277	6
2154	2278	6
2155	2279	5
2156	2280	6
2157	2281	6
2158	2282	6
2159	2283	6
2160	2284	5
2161	2285	5
2162	2286	5
2163	2287	4
2164	2288	4
2165	2289	4

```

[K_SMLINT_LRGINT, K_LRGINT_LRGINT]:
  BEGIN
    SELECTONE .CVT_PATH OF
      SET
        [K_SMLINT_LRGINT]:
          BEGIN
            M SCALE_L_OU;
          END;
        [K_LRGINT_LRGINT]:
          BEGIN
            M SCALE_OU_OU;
          END;
    TES;
    SELECTONE .DESTINATION [DSC$B_DTYPE] OF
      SET
        [DSC$K_DTYPE_LU]:
          BEGIN
            IF (.INTMED_DATA [LONG_2] OR .INTMED_DATA [LONG_3] OR .INTMED_DATA [LONG_4]) NEQ 0
              THEN SIGNAL (DBG$IINTOVF, 1, .DBG$GL_OPCODE_NAME);
            OUTPUT [LONG_1] = .INTMED_DATA [LONG_1];
            IF .SRC_INFO[S_SIGN]
              THEN
                OUTPUT[LONG_1] = -.OUTPUT[S_LONG_1];
              END;
          END;
        [DSC$K_DTYPE_Q, DSC$K_DTYPE_QU]:
          BEGIN
            IF (.INTMED_DATA [LONG_3] OR .INTMED_DATA [LONG_4]) NEQ 0
              THEN SIGNAL (DBG$IINTOVF, 1, .DBG$GL_OPCODE_NAME);
            IF .SRC_INFO[S_SIGN]
              THEN
                IF .INTMED_DATA [LONG_1] EQL 0
                  THEN
                    BEGIN
                      IF .INTMED_DATA [LONG_2] NEQU %X'80000000'
                        THEN
                          BEGIN
                            INTMED_DATA [LONG_2] = .INTMED_DATA [LONG_2] XOR %X'FFFFFFFF';
                            INTMED_DATA [LONG_2] = .INTMED_DATA [LONG_2] + 1;
                          END;
                        END
                    END
                  ELSE
                    BEGIN
                      INTMED_DATA [LONG_1] = .INTMED_DATA [LONG_1] XOR %X'FFFFFFFF';
                      INTMED_DATA [LONG_2] = .INTMED_DATA [LONG_2] XOR %X'FFFFFFFF';
                      INTMED_DATA [LONG_1] = .INTMED_DATA [LONG_1] + 1;
                    END;
                  END;
            OUTPUT [LONG_1] = .INTMED_DATA [LONG_1];
            OUTPUT [LONG_2] = .INTMED_DATA [LONG_2];
          END;
        [DSC$K_DTYPE_O]:

```

```

: 2166 2290 5
: 2167 2291 5
: 2168 2292 5
: 2169 2293 5
: 2170 2294 5
: 2171 2295 5
: 2172 2296 6
: 2173 2297 6
: 2174 2298 6
: 2175 2299 6
: 2176 2300 6
: 2177 2301 6
: 2178 2302 7
: 2179 2303 7
: 2180 2304 7
: 2181 2305 7
: 2182 2306 6
: 2183 2307 7
: 2184 2308 7
: 2185 2309 7
: 2186 2310 7
: 2187 2311 7
: 2188 2312 7
: 2189 2313 7
: 2190 2314 7
: 2191 2315 7
: 2192 2316 7
: 2193 2317 7
: 2194 2318 7
: 2195 2319 7
: 2196 2320 7
: 2197 2321 7
: 2198 2322 7
: 2199 2323 7
: 2200 2324 8
: 2201 2325 8
: 2202 2326 8
: 2203 2327 8
: 2204 2328 7
: 2205 2329 6
: 2206 2330 5
: 2207 2331 5
: 2208 2332 5
: 2209 2333 4
: 2210 2334 4
: 2211 2335 4
: 2212 2336 4
: 2213 2337 4
: 2214 2338 4
: 2215 2339 4
: 2216 2340 4
: 2217 2341 4
: 2218 2342 4
: 2219 2343 4
: 2220 2344 4
: 2221 2345 3

```

```

BEGIN
: The S_SIGN field is set if we need to negate the octaword.
IF .SRC_INFO[S_SIGN]
THEN
    BEGIN
        : Check for '80000000000000000000000000000000'
        : This should not go through the code below, but rather
        : just be deposited.
        IF NOT (.INTMED_DATA[LONG_1] EQL 0 AND
                .INTMED_DATA[LONG_2] EQL 0 AND
                .INTMED_DATA[LONG_3] EQL 0 AND
                .INTMED_DATA[LONG_4] EQL '80000000')
        THEN
            BEGIN
                MAP
                INTMED_DATA: VECTOR[K_INTMED_DATA_LENGTH/4];

                : Take the one's complement.
                INCR NEXT LONGWORD FROM 0 TO 3 DO
                    INTMED_DATA[NEXT_LONGWORD] =
                        .INTMED_DATA[NEXT_LONGWORD] XOR 'FFFFFFFF';

                : Add 1 to the result.
                INCR NEXT LONGWORD FROM 0 TO 3 DO
                    IF .INTMED_DATA[NEXT_LONGWORD] EQLU 'FFFFFFFF'
                    THEN
                        INTMED_DATA[NEXT_LONGWORD] = 0
                    ELSE
                        BEGIN
                            INTMED_DATA[NEXT_LONGWORD] =
                                .INTMED_DATA[NEXT_LONGWORD] + 1;
                        END;
                    EXITLOOP;
                END;
            END;
        END;

        CH$MOVE (16, INTMED_DATA, .OUTPUT);
        END;

[OTHERWISE]:
    SELECTONE .CVT_PATH OF
    SET
    [K_SMLINT_LRGINT]:
        $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: smlint_lrgint');

    [K_LRGINT_LRGINT]:
        $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: lrgint_lrgint');

    TES;
    !For SMLINT_LRGINT, LRGINT_LRGINT.
TES;
END;

```

```

2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279

```

```

[K_SMLINT_SMLFLTCMPLX, K_LRGINT_SMLFLTCMPLX, K_SMLFLTCMPLX_SMLFLTCMPLX, K_DEC_SMLFLTCMPLX, K_NBDS_SM
BEGIN
SELECTONE .CVT_PATH OF
SET

[K_SMLINT_SMLFLTCMPLX]:
BEGIN
M_SCALE_L_D;
END;

[K_LRGINT_SMLFLTCMPLX]:
BEGIN
M_SCALE_OU_D;
END;

[K_SMLFLTCMPLX_SMLFLTCMPLX]:
BEGIN
M_SCALE_D_D;
END;

[K_DEC_SMLFLTCMPLX]:
BEGIN
M_SCALE_P_D;
END;

[K_NBDS_SMLFLTCMPLX]:
BEGIN
CLASS_S_DESC [DSC$W_LENGTH] = .SRC_INFO [S_LEN];
CLASS_S_DESC [DSC$A_POINTER] = .SRC_INFO [S_POINTER];
STATUS = OT$CVT_T_D (CLASS_S_DESC, INTMED_DATA, 0, -.SCALE,
(K_IGN_BLK$ OR K_ENB_UNDERFLOW OR K_IGN_TAB$ OR K_ENB_SCALE));
IF NOT .STATUS THEN SIGNAL (DBG$_INVNUMSTR, 1, .DBG$GC_OPCODE_NAME);
END;
TES;

CASE .DESTINATION [DSC$B_DTYPE] FROM DSC$K_DTYPE_F TO DSC$K_DTYPE_D OF
SET

[DSC$K_DTYPE_F]:
CVTDF (INTMED_DATA, .OUTPUT);

[DSC$K_DTYPE_D]:
BEGIN
OUTPUT [LONG_1] = .INTMED_DATA [LONG_1];
OUTPUT [LONG_2] = .INTMED_DATA [LONG_2];
END;

[INRANGE, OUTRANGE]:
CASE .DESTINATION [DSC$B_DTYPE] FROM DSC$K_DTYPE_FC TO DSC$K_DTYPE_DC OF
SET

[DSC$K_DTYPE_FC]:
BEGIN
CVTDF (INTMED_DATA, .OUTPUT);
CVTDF (INTMED_DATA+8, .OUTPUT+4);
END;

```

2  
)  
1  
2  
4  
5  
8  
1  
2

DBGCVTDX  
V04-000

:	2280	2403	4
:	2281	2404	4
:	2282	2405	4
:	2283	2406	4
:	2284	2407	4
:	2285	2408	4
:	2286	2409	4
:	2287	2410	4
:	2288	2411	4
:	2289	2412	4
:	2290	2413	4
:	2291	2414	4
:	2292	2415	4
:	2293	2416	4
:	2294	2417	4
:	2295	2418	4
:	2296	2419	4
:	2297	2420	4
:	2298	2421	4
:	2299	2422	4
:	2300	2423	4
:	2301	2424	4
:	2302	2425	3

N 6  
15-Sep-1984 23:57:30 VAX-i1 Bliss-32 V4.0-742  
14-Sep-1984 12:16:44 [DEBUG.SRC]DBGCVTDX.B32;1

Page 55  
(11)

```
[DSC$K_DTYPE_DC]:  
  CH$MOVE 716, INTMED_DATA, .OUTPUT);  
[INRANGE, OUTRANGE]:  
  SELE$TONE .CVT_PATH OF  
    SET  
    [K_SMLINT_SMLFLTCMPLX]:  
      $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: smlint_smlfltcmplx');  
    [K_LRGINT_SMLFLTCMPLX]:  
      $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: lrgint_smlfltcmplx');  
    [K_SMLFLTCMPLX_SMLFLTCMPLX]:  
      $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: smlfltcmplx_smlfltcmplx');  
    [K_DEC_SMLFLTCMPLX]:  
      $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: dec_smlfltcmplx');  
    [K_NBDS_SMLFLTCMPLX]:  
      $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: nbds_smlfltcmplx');  
  TES;  
  TES;  
  TES;                                     !For SMLINT_SMLFLTCMPLX, LRGINT_SMLFLTCMPLX, SMLFLTCMPLX_SML  
IF .SRC_INFO [S_SIGN] THEN OUTPUT [0, 15, 1, 0] = 1;  
END;
```

2304 2426 3  
2305 2427 3  
2306 2428 4  
2307 2429 4  
2308 2430 4  
2309 2431 4  
2310 2432 4  
2311 2433 5  
2312 2434 5  
2313 2435 4  
2314 2436 4  
2315 2437 4  
2316 2438 5  
2317 2439 5  
2318 2440 4  
2319 2441 4  
2320 2442 4  
2321 2443 5  
2322 2444 5  
2323 2445 4  
2324 2446 4  
2325 2447 4  
2326 2448 5  
2327 2449 5  
2328 2450 5  
2329 2451 5  
2330 2452 6  
2331 2453 5  
2332 2454 5  
2333 2455 4  
2334 2456 4  
2335 2457 4  
2336 2458 5  
2337 2459 5  
2338 2460 4  
2339 2461 4  
2340 2462 4  
2341 2463 4  
2342 2464 4  
2343 2465 4  
2344 2466 4  
2345 2467 4  
2346 2468 4  
2347 2469 4  
2348 2470 4  
2349 2471 4  
2350 2472 4  
2351 2473 4  
2352 2474 4  
2353 2475 4  
2354 2476 4  
2355 2477 5  
2356 2478 5  
2357 2479 5  
2358 2480 4  
2359 2481 4  
2360 2482 4

```
[K_SMLINT_LRGFLTCMPLX, K_LRGINT_LRGFLTCMPLX, K_SMLFLTCMPLX_LRGFLTCMPLX, K_LRGFLTCMPLX_LRGFLTCMPLX, K
BEGIN
SELECTONE .CVT_PATH OF
SET

[K_SMLINT_LRGFLTCMPLX]:
BEGIN
M_SCALE_L_H;
END;

[K_LRGINT_LRGFLTCMPLX]:
BEGIN
M_SCALE_OU_H;
END;

[K_SMLFLTCMPLX_LRGFLTCMPLX]:
BEGIN
M_SCALE_D_H;
END;

[K_LRGFLTCMPLX_LRGFLTCMPLX]:
BEGIN
IF .SOURCE[DSC$B_DTYPE] EQL DSC$K_DTYPE_G OR
.SOURCE[DSC$B_DTYPE] EQL DSC$K_DTYPE_GC
THEN
M_SCALE_G_H
ELSE
M_SCALE_H_H;
END;

[K_DEC_LRGFLTCMPLX]:
BEGIN
M_SCALE_P_H;
END;
TES;

CASE .DESTINATION [DSC$B_DTYPE] FROM DSC$K_DTYPE_G TO DSC$K_DTYPE_H OF
SET

[DSC$K_DTYPE_G]:
CVTHG (INTMED_DATA, .OUTPUT);

[DSC$K_DTYPE_H]:
CH$MOVE (16, INTMED_DATA, .OUTPUT);

[INRANGE, OUTRANGE]:
CASE .DESTINATION[DSC$B_DTYPE] FROM DSC$K_DTYPE_GC TO DSC$K_DTYPE_HC OF
SET

[DSC$K_DTYPE_GC]:
BEGIN
CVTHG (INTMED_DATA, .OUTPUT);
CVTHG (INTMED_DATA+16, .OUTPUT+8);
END;

[DSC$K_DTYPE_HC]:
```

```

: 2361      2483  4
: 2362      2484  4
: 2363      2485  4
: 2364      2486  4
: 2365      2487  4
: 2366      2488  4
: 2367      2489  4
: 2368      2490  4
: 2369      2491  4
: 2370      2492  4
: 2371      2493  4
: 2372      2494  4
: 2373      2495  4
: 2374      2496  4
: 2375      2497  4
: 2376      2498  4
: 2377      2499  4
: 2378      2500  4
: 2379      2501  4
: 2380      2502  4
: 2381      2503  3

```

```

CHSMOVE (32, INTMED_DATA, .OUTPUT);
[INRANGE, OUTRANGE]:
  SELECTONE .CVT_PATH OF
  SET
  [K_SMLINT_LRGFLTCMPLX]:
    $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: smlint_lrgfltcmplx');
  [K_LRGINT_LRGFLTCMPLX]:
    $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: lrgint_lrgfltcmplx');
  [K_SMLFLTCMPLX_LRGFLTCMPLX]:
    $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: smlfltcmplx_lrgfltcmplx');
  [K_LRGFLTCMPLX_LRGFLTCMPLX]:
    $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: lrgfltcmplx_lrgfltcmplx');
  [K_DEC_LRGFLTCMPLX]:
    $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: dec_lrgfltcmplx');
  TES;
  TES;
  TES; !For SMLINT_LRGFLTCMPLX, LRGINT_LRGFLTCMPLX, SMLFLTCMPLX_LRGFLTCMPLX, LRGFLT
IF .SRC_INFO [S_SIGN] THEN OUTPUT [0, 15, 1, 0] = 1;
END;

```

2383	2504	3
2384	2505	3
2385	2506	4
2386	2507	4
2387	2508	4
2388	2509	4
2389	2510	4
2390	2511	5
2391	2512	5
2392	2513	4
2393	2514	4
2394	2515	4
2395	2516	5
2396	2517	5
2397	2518	4
2398	2519	4
2399	2520	4
2400	2521	4
2401	2522	4
2402	2523	4
2403	2524	4
2404	2525	5
2405	2526	5
2406	2527	5
2407	2528	4
2408	2529	4
2409	2530	4
2410	2531	4
2411	2532	4
2412	2533	4
2413	2534	4
2414	2535	4
2415	2536	4
2416	2537	5
2417	2538	5
2418	2539	7
2419	2540	5
2420	2541	5
2421	2542	4
2422	2543	4
2423	2544	4
2424	2545	5
2425	2546	5
2426	2547	5
2427	2548	5
2428	2549	6
2429	2550	6
2430	2551	5
2431	2552	5
2432	2553	5
2433	2554	5
2434	2555	5
2435	2556	4
2436	2557	4
2437	2558	4
2438	2559	4
2439	2560	5

```

[K_SMLINT_DEC, K_DEC_DEC]:
  BEGIN
    SELECTONE .CVT_PATH OF
      SET
        [K_SMLINT_DEC]:
          BEGIN
            M_SCALE_L_P;
          END;
        [K_DEC_DEC]:
          BEGIN
            M_SCALE_P_P;
          END;
      TES;
    CASE .DESTINATION [DSC$B_DTYPE] FROM DSC$K_DTYPE_NU TO DSC$K_DTYPE_P OF
      SET
        [DSC$K_DTYPE_NU]:
          BEGIN
            IF .SRC_INFO [S_SIGN] THEN SIGNAL (DBG$ CVTNEGUNS, 1, .DBG$GL_OPCODE_NAME);
            CVTPT (NO_DIGITS, INTMED_DATA, LIB$AB_CVTPT_U, DESTINATION [DSC$W_LENGTH], .OUTPUT);
          END;
        [DSC$K_DTYPE_NL]:
          CVTPT (NO_DIGITS, INTMED_DATA,
            %REF ?
            IF .DESTINATION [DSC$W_LENGTH] EQL 0 THEN 0 ELSE .DESTINATION [DSC$W_LENGTH] - 1
            , .OUTPUT);
        [DSC$K_DTYPE_NLO]:
          BEGIN
            CVTPT (NO_DIGITS, INTMED_DATA, LIB$AB_CVTPT_U, DESTINATION [DSC$W_LENGTH], TEMP_BUF1);
            TEMP_BUF1 [BYTE_1] = (IF .SRC_INFO [S_SIGN] THEN (.TEMP_BUF1 [BYTE_1] + LIB$AB_CVT_U_0
              48 + 10) ELSE (.TEMP_BUF1 [BYTE_1] + LIB$AB_CVT_U_0 - 48));
            CH$MOVE (.DESTINATION [DSC$W_LENGTH], TEMP_BUF1, .OUTPUT);
          END;
        [DSC$K_DTYPE_NR]:
          BEGIN
            LOCAL
              DES_LEN;
            DES_LEN =
              BEGIN
                IF .DESTINATION [DSC$W_LENGTH] EQL 0 THEN 0 ELSE .DESTINATION [DSC$W_LENGTH] - 1
              END;
            CVTPT (NO_DIGITS, INTMED_DATA, DES_LEN, TEMP_BUF1);
            BLOCK [INTMED_DATA + .DES_LEN, 0, 0, 8, 0, .BYTE] = .TEMP_BUF1 [BYTE_1];
            CH$MOVE (.DES_LEN, TEMP_BUF1 + 1, INTMED_DATA);
            CH$MOVE (.DES_LEN + 1, INTMED_DATA, .OUTPUT);
          END;
        [DSC$K_DTYPE_NRO, DSC$K_DTYPE_NZ]:
          CVTPT (NO_DIGITS, INTMED_DATA,
            (IF .DESTINATION [DSC$B_DTYPE] EQL DSC$K_DTYPE_NRO THEN LIB$AB_CVTPT_O ELSE

```



```

: 2440      2561  4
: 2441      2562  4
: 2442      2563  4
: 2443      2564  5
: 2444      2565  5
: 2445      2566  5
: 2446      2567  5
: 2447      2568  5
: 2448      2569  4
: 2449      2570  4
: 2450      2571  4
: 2451      2572  4
: 2452      2573  4
: 2453      2574  4
: 2454      2575  4
: 2455      2576  4
: 2456      2577  4
: 2457      2578  4
: 2458      2579  4
: 2459      2580  3

```

```

LIB$AB_CVTPT_2), DESTINATION [DSC$W_LENGTH], .OUTPUT);

[DSC$K_DTYPE_P]:
BEGIN
  CVTSP (NO_DIGITS, INTMED_DATA, DESTINATION [DSC$W_LENGTH], TEMP_BUF1);
  CVTSP (DESTINATION [DSC$W_LENGTH], TEMP_BUF1, DESTINATION [DSC$W_LENGTH], .OUTPUT);
  CVTSP (NO_DIGITS, INTMED_DATA, NO_DIGITS, TEMP_BUF1);
  CVTSP (NO_DIGITS, TEMP_BUF1, DESTINATION [DSC$W_LENGTH], .OUTPUT);
END;

[INRANGE, OVRANGE]:
SELECT ONE .CVT_PATH OF
  SET
  [K_SMLINT_DEC]:
    $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: smlint_dec');
  [K_DEC_DEC]:
    $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: dec_dec');
  TES;
TES;
!For SMLINT_DEC, DEC_DEC
END;

```

2461 2581 3  
2462 2582 3  
2463 2583 4  
2464 2584 4  
2465 2585 4  
2466 2586 4  
2467 2587 4  
2468 2588 4  
2469 2589 4  
2470 2590 4  
2471 2591 4  
2472 2592 5  
2473 2593 5  
2474 2594 5  
2475 2595 5  
2476 2596 4  
2477 2597 4  
2478 2598 4  
2479 2599 5  
2480 2600 5  
2481 2601 5  
2482 2602 4  
2483 2603 4  
2484 2604 4  
2485 2605 5  
2486 2606 5  
2487 2607 5  
2488 2608 5  
2489 2609 4  
2490 2610 4  
2491 2611 4  
2492 2612 5  
2493 2613 5  
2494 2614 5  
2495 2615 5  
2496 2616 4  
2497 2617 4  
2498 2618 4  
2499 2619 5  
2500 2620 5  
2501 2621 5  
2502 2622 5  
2503 2623 5  
2504 2624 6  
2505 2625 6  
2506 2626 6  
2507 2627 6  
2508 2628 5  
2509 2629 4  
2510 2630 4  
2511 2631 4  
2512 2632 5  
2513 2633 5  
2514 2634 5  
2515 2635 5  
2516 2636 5  
2517 2637 5

```
[K_LRGINT_SMLINT]:
BEGIN
M_SCALE OU OU;
IF (.INTMED_DATA [LONG_2] OR .INTMED_DATA [LONG_3] OR .INTMED_DATA [LONG_4]) NEQ 0
THEN
    SIGNAL (DBG$IINTOVF, 1, .DBG$GL_OPCODE_NAME);
CASE .DESTINATION [DSC$B_DTYPE] FROM DSC$K_DTYPE_V TO DSC$K_DTYPE_SVU OF
    SET
        [DSC$K_DTYPE_BU]:
        BEGIN
            IF .INTMED_DATA [BYTE_2] OR .INTMED_DATA [WORD_2] NEQ 0
            THEN SIGNAL (DBG$IINTOVF, 1, .DBG$GL_OPCODE_NAME);
            OUTPUT [BYTE_1] = .INTMED_DATA [LONG_1];
        END;
        [DSC$K_DTYPE_WU]:
        BEGIN
            IF .INTMED_DATA [WORD_2] NEQ 0 THEN SIGNAL (DBG$IINTOVF, 1, .DBG$GL_OPCODE_NAME);
            OUTPUT [WORD_1] = .INTMED_DATA [LONG_1];
        END;
        [DSC$K_DTYPE_B]:
        BEGIN
            IF .INTMED_DATA [S_LONG_1] LSS 0 THEN SIGNAL (DBG$IINTOVF, 1, .DBG$GL_OPCODE_NAME);
            IF .SRC_INFO [S_SIGN] THEN INTMED_DATA [LONG_1] = -.INTMED_DATA [S_LONG_1];
            CVTLB (INTMED_DATA, .OUTPUT);
        END;
        [DSC$K_DTYPE_W]:
        BEGIN
            IF .INTMED_DATA [S_LONG_1] LSS 0 THEN SIGNAL (DBG$IINTOVF, 1, .DBG$GL_OPCODE_NAME);
            IF .SRC_INFO [S_SIGN] THEN INTMED_DATA [LONG_1] = -.INTMED_DATA [S_LONG_1];
            CVTLW (INTMED_DATA, .OUTPUT);
        END;
        [DSC$K_DTYPE_L]:
        BEGIN
            IF .INTMED_DATA [S_LONG_1] EQL K_LRGST_NEG_L AND .SRC_INFO [S_SIGN] EQL 1
            THEN
                OUTPUT [LONG_1] = .INTMED_DATA [S_LONG_1]
            ELSE
                BEGIN
                    IF .INTMED_DATA [S_LONG_1] LSS 0 THEN SIGNAL (DBG$IINTOVF, 1, .DBG$GL_OPCODE_NAME);
                    IF .SRC_INFO [S_SIGN] THEN INTMED_DATA [LONG_1] = -.INTMED_DATA [S_LONG_1];
                    OUTPUT [LONG_1] = .INTMED_DATA [S_LONG_1];
                END;
            END;
        [DSC$K_DTYPE_V, DSC$K_DTYPE_SV, DSC$K_DTYPE_VU, DSC$K_DTYPE_SVU, DSC$K_DTYPE_TF]:
        BEGIN
            MAP
                OUTPUT: REF BITVECTOR[K_OUTPUT_BUFFER_LENGTH * 8],
                INTMED_DATA: BITVECTOR[K_INTMED_DATA_LENGTH * 8];
            INCR I FROM 0 TO .DST_INFO[D_LEN] - 1 DO
```

DBGCVTDX  
V04-000

G 7  
13-Sep-1984 23:57:30 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:16:44 [DEBUG.SRC]DBGCVTDX.B32;1

Page 61  
(14)

: 2518	2638	6
: 2519	2639	6
: 2520	2640	5
: 2521	2641	4
: 2522	2642	4
: 2523	2643	4
: 2524	2644	4
: 2525	2645	4
: 2526	2646	3

```
      BEGIN
      OUTPUT[.I] = .INTMED_DATA[.I];
      END;
    END;
  [INRANGE, OTRANGE]:
  SDBG_ERROR ('DBGCVTDX\DBG$CVT DX-DX: lrgint smlint');
  TES;
  !For LRGINT_SMLINT
END;
```

2528 2647 3  
2529 2648 3  
2530 2649 4  
2531 2650 4  
2532 2651 4  
2533 2652 4  
2534 2653 4  
2535 2654 4  
2536 2655 4  
2537 2656 5  
2538 2657 5  
2539 2658 5  
2540 2659 5  
2541 2660 4  
2542 2661 4  
2543 2662 4  
2544 2663 5  
2545 2664 5  
2546 2665 5  
2547 2666 4  
2548 2667 4  
2549 2668 4  
2550 2669 5  
2551 2670 5  
2552 2671 5  
2553 2672 5  
2554 2673 5  
2555 2674 5  
2556 2675 5  
2557 2676 5  
2558 2677 4  
2559 2678 4  
2560 2679 4  
2561 2680 5  
2562 2681 5  
2563 2682 5  
2564 2683 5  
2565 2684 5  
2566 2685 5  
2567 2686 5  
2568 2687 5  
2569 2688 5  
2570 2689 5  
2571 2690 4  
2572 2691 4  
2573 2692 4  
2574 2693 4  
2575 2694 4  
2576 2695 4  
2577 2696 4  
2578 2697 4  
2579 2698 4  
2580 2699 4  
2581 2700 4  
2582 2701 5  
2583 2702 5  
2584 2703 5

```
[K_LRGINT_DEC, K_SMLFLTCMPLX_DEC, K_LRGFLTCMPLX_DEC, K_NBDS_DEC]:
BEGIN
  CLASS_S_DESC [DSC$W_LENGTH] = K_TEMP_BUF_LENGTH;
  CLASS_S_DESC [DSC$A_POINTER] = TEMP_BUF2;
  SELECTONE .CVT_PATH OF
  SET

  [K_LRGINT_DEC]:
  BEGIN
    C.TROUH (INTMED_DATA, TEMP_BUF1);
    IF .SRC_INFO [S_SIGN] THEN TEMP_BUF1<15, 1, 0> = 1;
    STATUS = FOR$CVT_H_TF (TEMP_BUF1, CLASS_S_DESC, 0, .SCALE, 0, 0, 1);
  END;

  [K_SMLFLTCMPLX_DEC]:
  BEGIN
    IF .INTMED_DATA<15, 1, 0> THEN SRC_INFO [S_SIGN] = 1;
    STATUS = FOR$CVT_D_TF (INTMED_DATA, CLASS_S_DESC, 0, .SCALE, 0, 0, 1);
  END;

  [K_LRGFLTCMPLX_DEC]:
  BEGIN
    IF .INTMED_DATA<15, 1, 0> THEN SRC_INFO [S_SIGN] = 1;
    IF .SOURCE[DSC$B_DTYPE] EQL DSC$K_DTYPE_G OR
    .SOURCE[DSC$B_DTYPE] EQL DSC$K_DTYPE_GC
    THEN
      STATUS = FOR$CVT_G_TF (INTMED_DATA, CLASS_S_DESC, 0, .SCALE, 0, 0, 1)
    ELSE
      STATUS = FOR$CVT_H_TF (INTMED_DATA, CLASS_S_DESC, 0, .SCALE, 0, 0, 1);
    END;

  [K_NBDS_DEC]:
  BEGIN
    CLASS_S_DESC [DSC$W_LENGTH] = .SRC_INFO [S_LEN];
    CLASS_S_DESC [DSC$A_POINTER] = .SRC_INFO [S_POINTER];
    STATUS = OT$CVT_T_H (CLASS_S_DESC, TEMP_BUF1, 0, -.SCALE,
    (K_IGN_BLK$ OR K_ENB_UNDERFLOW OR K_IGN_TAB$ OR K_ENB_SCALE));
    IF NOT .STATUS THEN SIGNAL (DBG$ INVNUMSTR, 1, .DBG$GL_OPCODE_NAME);
    IF .TEMP_BUF1<15, 1, 0> THEN SRC_INFO [S_SIGN] = 1;
    CLASS_S_DESC [DSC$W_LENGTH] = K_TEMP_BUF_LENGTH;
    CLASS_S_DESC [DSC$A_POINTER] = TEMP_BUF2;
    STATUS = FOR$CVT_H_TF (TEMP_BUF1, CLASS_S_DESC, 0, 0, 0, 0, 1);
  END;

  TES;

  IF NOT .STATUS THEN SIGNAL (DBG$ DECOVF, 1, .DBG$GL_OPCODE_NAME);
  BUF_OFFSET = CH$FIND NOT CH (K_TEMP_BUF_LENGTH, TEMP_BUF2, 'XC' ) - TEMP_BUF2;
  NO_DIGITS = K_TEMP_BUF_LENGTH - .BUF_OFFSET - 2;

  CASE .DESTINATION [DSC$B_DTYPE] FROM DSC$K_DTYPE_NU TO DSC$K_DTYPE_P OF
  SET

  [DSC$K_DTYPE_NU]:
  BEGIN
    IF .SRC_INFO [S_SIGN] THEN SIGNAL (DBG$ CVTNEGUNS, 1, .DBG$GL_OPCODE_NAME);
    IF .NO_DIGITS GTR .DESTINATION [DSC$W_LENGTH] THEN SIGNAL (DBG$ DECOVF, 1, .DBG$GL_OPCODE
```

```

: 2585      2704      5      CH$FILL (XX'30', .DESTINATION [DSC$W_LENGTH] - .NO_DIGITS, TEMP_BUF1);
: 2586      2705      5      CH$MOVE (.NO_DIGITS, TEMP_BUF2 + .BUF_OFFSET + 1,
: 2587      2706      5      TEMP_BUF1 + .DESTINATION [DSC$W_LENGTH] - .NO_DIGITS);
: 2588      2707      5      CH$MOVE (.DESTINATION [DSC$W_LENGTH], TEMP_BUF1, .OUTPUT);
: 2589      2708      4      END;
: 2590      2709      4
: 2591      2710      4      [DSC$K_DTYPE_NL]:
: 2592      2711      5      BEGIN
: 2593      2712      5      LOCAL
: 2594      2713      5      DES_LEN;
: 2595      2714      5      DES_LEN =
: 2596      2715      6      BEGIN
: 2597      2716      6      IF .DESTINATION [DSC$W_LENGTH] EQL 0 THEN 0 ELSE .DESTINATION [DSC$W_LENGTH] - 1
: 2598      2717      5      END;
: 2599      2718      5      IF .DES_LEN LSS .NO_DIGITS THEN SIGNAL (DBG$ DECOVF, 1, .DBG$GL_OPCODE_NAME);
: 2600      2719      5      CVTSP (.NO_DIGITS, TEMP_BUF2 + .BUF_OFFSET, DES_LEN, TEMP_BUF1);
: 2601      2720      5      CVTSP (DES_LEN, TEMP_BUF1, DES_LEN, .OUTPUT);
: 2602      2721      4      END;
: 2603      2722      4
: 2604      2723      4      [DSC$K_DTYPE_NLO]:
: 2605      2724      5      BEGIN
: 2606      2725      5      CH$FILL (XX'30', .BUF_OFFSET + 1, TEMP_BUF2);
: 2607      2726      5      IF .NO_DIGITS GTR .DESTINATION [DSC$W_LENGTH] THEN SIGNAL (DBG$ DECOVF, 1, .DBG$GL_OPCODE
: 2608      2727      5      BUF_OFFSET = K TEMP_BUF_LENGTH - .DESTINATION [DSC$W_LENGTH] - 1;
: 2609      2728      7      BLOCK [TEMP_BUF2 + .BUF_OFFSET, 0, 0, 8, 0; .BYTE] = (IF .SRC_INFO [S_SIGN] THEN (.BLOC
: 2610      2729      7      [TEMP_BUF2 + .BUF_OFFSET, 0, 0, 8, 0; .BYTE] + LIB$AB_CVT_U_0 - 48 + 10) ELSE (.
: 2611      2730      5      .BLOC [TEMP_BUF2 + .BUF_OFFSET, 0, 0, 8, 0; .BYTE] + LIB$AB_CVT_U_0 - 48));
: 2612      2731      5      CH$MOVE (.DESTINATION [DSC$W_LENGTH], TEMP_BUF2 + .BUF_OFFSET, .OUTPUT);
: 2613      2732      4      END;
: 2614      2733      4
: 2615      2734      4      [DSC$K_DTYPE_NR]:
: 2616      2735      5      BEGIN
: 2617      2736      5      LOCAL
: 2618      2737      5      DES_LEN;
: 2619      2738      5      DES_LEN =
: 2620      2739      6      BEGIN
: 2621      2740      6      IF .DESTINATION [DSC$W_LENGTH] EQL 0 THEN 0 ELSE .DESTINATION [DSC$W_LENGTH] - 1
: 2622      2741      5      END;
: 2623      2742      5      IF .NO_DIGITS GTR .DES_LEN THEN SIGNAL (DBG$ DECOVF, 1, .DBG$GL_OPCODE_NAME);
: 2624      2743      5      CH$FILL (XX'30', .DES_LEN - .NO_DIGITS + 1, TEMP_BUF1);
: 2625      2744      5      CH$MOVE (.NO_DIGITS, TEMP_BUF2 + .BUF_OFFSET + 1, TEMP_BUF1 + .DES_LEN - .NO_DIGITS);
: 2626      2745      5      BLOCK [TEMP_BUF1 + .DES_LEN, 0, 0, 8, 0; .BYTE] = .BLOCK [TEMP_BUF2 + .BUF_OFFSET, 0,
: 2627      2746      5      0, 8, 0; .BYTE];
: 2628      2747      5      CH$MOVE (.DES_LEN + 1, TEMP_BUF1, .OUTPUT);
: 2629      2748      4      END;
: 2630      2749      4
: 2631      2750      4      [DSC$K_DTYPE_NRO, DSC$K_DTYPE_NZ]:
: 2632      2751      5      BEGIN
: 2633      2752      5      IF .NO_DIGITS GTR .DESTINATION [DSC$W_LENGTH] THEN SIGNAL (DBG$ DECOVF, 1, .DBG$GL_OPCODE
: 2634      2753      5      CVTSP (.NO_DIGITS, TEMP_BUF2 + .BUF_OFFSET, DESTINATION [DSC$W_LENGTH], TEMP_BUF1);
: 2635      2754      5      CVTPT (DESTINATION [DSC$W_LENGTH], TEMP_BUF1,
: 2636      2755      6      (IF .DESTINATION [DSC$B_DTYPE] EQL DSC$K_DTYPE_NRO THEN LIB$AB_CVTPT_0 ELSE
: 2637      2756      5      LIB$AB_CVTPT_2), DESTINATION [DSC$W_LENGTH], .OUTPUT);
: 2638      2757      4      END;
: 2639      2758      4
: 2640      2759      4      [DSC$K_DTYPE_P]:
: 2641      2760      5      BEGIN

```

DBGCVTDX  
V04-000

J 7  
15-Sep-1984 23:57:30 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:16:44 [DEBUG.SRC]DBGCVTDX.B32;1

Page 64  
(15)

: 2642 2761 5  
: 2643 2762 5  
: 2644 2763 4  
: 2645 2764 4  
: 2646 2765 4  
: 2647 2766 4  
: 2648 2767 4  
: 2649 2768 4  
: 2650 2769 4  
: 2651 2770 4  
: 2652 2771 4  
: 2653 2772 4  
: 2654 2773 4  
: 2655 2774 4  
: 2656 2775 4  
: 2657 2776 4  
: 2658 2777 4  
: 2659 2778 3

```
IF .NO DIGITS GTR 31 THEN SIGNAL (DBG$ DECOVF, 1, .DBG$GL_OPCODE NAME);
CVTSP TNO_DIGITS, TEMP_BUF2 + .BUF_OFFSET, DESTINATION [DSC$W_LENGTH], .OUTPUT);
END;

[INRANGE, OTRANGE]:
SELECTONE .CVT_PATH OF
SET
[K_LRGINT_DEC]:
    $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: lrgint_dec');
[K_SMLFLTCMPLX_DEC]:
    $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: smlfltcmplx_dec');
[K_LRGFLTCMPLX_DEC]:
    $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: lrgfltcmplx_dec');
[K_NBDS_DEC]:
    $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: nbds_dec');
TES;
TES;
END; !For LRGINT_DEC, SMLFLTCMPLX_DEC, LRGFLTCMPLX_DEC, NBDS_DEC.
```

2661 2779 3  
2662 2780 3  
2663 2781 3  
2664 2782 3  
2665 2783 3  
2666 2784 3  
2667 2785 4  
2668 2786 4  
2669 2787 4  
2670 2788 4  
2671 2789 4  
2672 2790 4  
2673 2791 4  
2674 2792 4  
2675 2793 4  
2676 2794 4  
2677 2795 4  
2678 2796 4  
2679 2797 4  
2680 2798 4  
2681 2799 4  
2682 2800 4  
2683 2801 5  
2684 2802 4  
2685 2803 4  
2686 2804 4  
2687 2805 4  
2688 2806 4  
2689 2807 4  
2690 2808 4  
2691 2809 4  
2692 2810 4  
2693 2811 4  
2694 2812 4  
2695 2813 4  
2696 2814 4  
2697 2815 5  
2698 2816 5  
2699 2817 5  
2700 2818 5  
2701 2819 5  
2702 2820 5  
2703 2821 5  
2704 2822 6  
2705 2823 6  
2706 2824 6  
2707 2825 5  
2708 2826 5  
2709 2827 6  
2710 2828 6  
2711 2829 6  
2712 2830 5  
2713 2831 5  
2714 2832 5  
2715 2833 4  
2716 2834 4  
2717 2835 4

```
[K_SMLINT_NBDS, K_LRGINT_NBDS, K_DEC_NBDS]:
  SELECTONE .DESTINATION [DSC$B_DTYPE] OF
  SET
    [DSC$K_DTYPE_BU, DSC$K_DTYPE_T, DSC$K_DTYPE_VT, DSC$K_DTYPE_AC, DSC$K_DTYPE_AZ]:
    BEGIN
      CLASS_S_DESC [DSC$W_LENGTH] = K_TEMP_BUF_LENGTH;
      CLASS_S_DESC [DSC$A_POINTER] = TEMP_BUF2;

      ! Compute 'DIGITS_IN_FRACT' based on scale.
      ! For negative scales, the number of digits in the fraction
      ! is just the absolute value of the scale. This seems
      ! to work for both binary and decimal scales. For example,
      ! (binary 101 with scale factor -2) = binary 1.01 =
      ! 1 + 0/2 + 1/4 = 1.25, which has 2 digits in the fraction.
      ! For non-negative scale, DIGITS_IN_FRACT is zero.
      ! First do a consistency check to ensure we do not have
      ! both decimal and binary scale factors - if we do,
      ! something is wrong.
      DIGITS_IN_FRACT = 0;
      IF (.BIN_SCALE NEQ 0) AND (.SCALE NEQ 0)
      THEN
        $DBG_ERROR('DBGCVTDX\DBG$CVT_DX_DX inconsistent scale factors');
      IF .BIN_SCALE LSS 0
      THEN
        DIGITS_IN_FRACT = -.BIN_SCALE;
      IF .SCALE LSS 0
      THEN
        DIGITS_IN_FRACT = -.SCALE;
      SELECTONE .CVT_PATH OF
      SET
        [K_SMLINT_NBDS]:
        BEGIN
          CVTLD (INTMED_DATA, TEMP_BUF1);

          ! Take care of binary scale factors by doing
          ! the divide or multiply.
          WHILE .BIN_SCALE LSS 0 DO
            BEGIN
              DIVD2(UPLIT (%D'2.0'), TEMP_BUF1);
              BIN_SCALE = .BIN_SCALE + 1;
            END;
          WHILE .BIN_SCALE GTR 0 DO
            BEGIN
              MULD2(UPLIT (%D'2.0'), TEMP_BUF1);
              BIN_SCALE = .BIN_SCALE - 1;
            END;

          STATUS = FOR$CVT_D_TF (TEMP_BUF1, CLASS_S_DESC, .DIGITS_IN_FRACT, .SCALE);
          END;
        [K_LRGINT_NBDS]:
```

```

2718 2836 4
2719 2837 4
2720 2838 5
2721 2839 5
2722 2840 5
2723 2841 5
2724 2842 5
2725 2843 5
2726 2844 5
2727 2845 5
2728 2846 5
2729 2847 6
2730 2848 6
2731 2849 6
2732 2850 5
2733 2851 5
2734 2852 6
2735 2853 6
2736 2854 6
2737 2855 5
2738 2856 5
2739 2857 5
2740 2858 5
2741 2859 4
2742 2860 5
2743 2861 5
2744 2862 5
2745 2863 5
2746 2864 5
2747 2865 5
2748 2866 5
2749 2867 5
2750 2868 5
2751 2869 5
2752 2870 5
2753 2871 5
2754 2872 5
2755 2873 5
2756 2874 5
2757 2875 5
2758 2876 5
2759 2877 5
2760 2878 5
2761 2879 5
2762 2880 5
2763 2881 5
2764 2882 5
2765 2883 5
2766 2884 5
2767 2885 5
2768 2886 5
2769 2887 5
2770 2888 5
2771 2889 5
2772 2890 5
2773 2891 6
2774 2892 6

```

```

IF .SOURCE[ DSC$B_DTYPE ] NEQ DSC$K_DTYPE_0      ! A004
THEN                                              ! A004
BEGIN
  CVTROUH (INTMED_DATA, TEMP_BUF1);
  IF .SRC_INFO [S_SIGN] THEN TEMP_BUF1<15, 1, 0> = 1;

  ! Take care of binary scale factors by doing
  ! the divide or multiply.
  WHILE .BIN_SCALE LSS 0 DO
    BEGIN
      DIVH2(UPLIT (XH'2.0'), TEMP_BUF1);
      BIN_SCALE = .BIN_SCALE + 1;
    END;
  WHILE .BIN_SCALE GTR 0 DO
    BEGIN
      MULH2(UPLIT (XH'2.0'), TEMP_BUF1);
      BIN_SCALE = .BIN_SCALE - 1;
    END;

  STATUS = FOR$CVT_H_TF (TEMP_BUF1, CLASS_S_DESC, .DIGITS_IN_FRACT, .SCALE);
END
ELSE
  BEGIN
    LOCAL
      Previous_Value : VECTOR[4];

    MAP
      INTMED_DATA : VECTOR[4];

    ! Don't support scale factor on octaword.
    IF .BIN_SCALE NEQ 0
    THEN
      $DBG_ERROR('DBGCVTDX\DBG$CVT_DX_DX scale factor on octaword not supporte

  CLASS_S_DESC[ DSC$W_LENGTH ] = 0;      ! A004

  !++
  ! Init the Previous value
  !--
  CH$MOVE( 16,
    CH$PTR( INTMED_DATA),
    CH$PTR( Previous_value ) );
    ! A004
    ! A004
    ! A004

  !++
  ! By dividing the value by ten and multiplying it by
  ! ten the original value and the new value may be
  ! subtracted to obtain the value of the least
  ! significant digit.
  ! Repeating allows the building up of the string
  ! from the back.
  !--
DO
  BEGIN
    LOCAL

```



2775	2893	6
2776	2894	6
2777	2895	6
2778	2896	6
2779	2897	6
2780	2898	6
2781	2899	6
2782	2900	6
2783	2901	6
2784	2902	6
2785	2903	6
2786	2904	6
2787	2905	6
2788	2906	6
2789	2907	6
2790	2908	6
2791	2909	6
2792	2910	6
2793	2911	6
2794	2912	6
2795	2913	6
2796	2914	6
2797	2915	6
2798	2916	6
2799	2917	6
2800	2918	6
2801	2919	6
2802	2920	6
2803	2921	6
2804	2922	6
2805	2923	6
2806	2924	6
2807	2925	6
2808	2926	6
2809	2927	6
2810	2928	6
2811	2929	6
2812	2930	6
2813	2931	6
2814	2932	6
2815	2933	6
2816	2934	6
2817	2935	6
2818	2936	6
2819	2937	6
2820	2938	6
2821	2939	6
2822	2940	6
2823	2941	6
2824	2942	6
2825	2943	6
2826	2944	6
2827	2945	6
2828	2946	6
2829	2947	5
2830	2948	5
2831	2949	5

```

Saved_Value : VECTOR[4];                                ! A004

++
Save the previous value
--
CH$MOVE( 16,                                             ! A004
        CH$PTR( INTMED_DATA ),                          ! A004
        CH$PTR( Previous_value ) );                    ! A004

++
Divide by ten
--
DBG$CVT_SCALE_OU_DOWN_BY_10_R1( INTMED_DATA );          ! A004

++
Save the divided value for the next time
--
CH$MOVE( 16,                                             ! A004
        CH$PTR( INTMED_DATA ),                          ! A004
        CH$PTR( Saved_value ) );                      ! A004

++
Multiply by ten to remove for the subtraction
--
DBG$CVT_SCALE_OU_UP_BY_10_R1( INTMED_DATA );            ! A004

++
Move the previous digits down
--
DECR Current_position FROM .CLASS_S_DESC[DSC$W_LENGTH] - 1
TO 0 DO
    CH$WCHAR( CH$RCHAR( CH$PTR( .CLASS_S_DESC[ DSC$A_POINTER ] + .Current_position
    CH$PTR( .CLASS_S_DESC[ DSC$A_POINTER ] + .Current_position

++
Subtract and put the new digit in the string
--
CH$WCHAR( .Previous_value[0] - .INTMED_DATA[0] + %C'0', ! A004
        CH$PTR( .CLASS_S_DESC[ DSC$A_POINTER ] ) );    ! A004

++
Increment the length
--
CLASS_S_DESC[ DSC$W_LENGTH ] =                          ! A004
    .CLASS_S_DESC[ DSC$W_LENGTH ] + 1;                  ! A004

++
Saved value becomes the previous value
--
CH$MOVE( 16,                                             ! A004
        CH$PTR( Saved_value ),                          ! A004
        CH$PTR( INTMED_DATA ) );                      ! A004

END                                                     ! A004
WHILE ( .INTMED_DATA[ 3 ] NEQ 0 ) OR                   ! A004
      ( .INTMED_DATA[ 2 ] NEQ 0 ) OR                   ! A004
      ( .INTMED_DATA[ 1 ] NEQ 0 ) OR                   ! A004

```

2832 2950 5  
2833 2951 5  
2834 2952 5  
2835 2953 5  
2836 2954 5  
2837 2955 5  
2838 2956 5  
2839 2957 6  
2840 2958 6  
2841 2959 6  
2842 2960 6  
2843 2961 6  
2844 2962 6  
2845 2963 6  
2846 2964 6  
2847 2965 5  
2848 2966 5  
2849 2967 5  
2850 2968 5  
2851 2969 5  
2852 2970 5  
2853 2971 5  
2854 2972 5  
2855 2973 5  
2856 2974 4  
2857 2975 4  
2858 2976 4  
2859 2977 5  
2860 2978 5  
2861 2979 5  
2862 2980 5  
2863 2981 5  
2864 2982 5  
2865 2983 5  
2866 2984 5  
2867 2985 5  
2868 2986 5  
2869 2987 5  
2870 2988 5  
2871 2989 5  
2872 2990 5  
2873 2991 4  
2874 2992 4  
2875 2993 4  
2876 2994 4  
2877 2995 4  
2878 2996 4  
2879 2997 4  
2880 2998 4  
2881 2999 4  
2882 3000 4  
2883 3001 4  
2884 3002 4  
2885 3003 4  
2886 3004 4  
2887 3005 4  
2888 3006 4

```
(.INTMED_DATA[ 0 ] NEQ 0);                                ! A004
                                                                ! A004
++
--
Load in a '-' if there is one
IF .SRC_INFO[ S_SIGN ]
THEN
BEGIN
  DECR Current_position FROM .CLASS_S_DESC[DSC$W_LENGTH] - 1
  TO 0 DO
    CH$WCHAR( CH$RCHAR( CH$PTR( .CLASS_S_DESC[ DSC$A_POINTER ] + .Current_position
    CH$PTR( .CLASS_S_DESC[ DSC$A_POINTER ] + .Current_position
    CH$WCHAR( %C'-'
    CH$PTR( .CLASS_S_DESC[ DSC$A_POINTER ] ) );
  CLASS_S_DESC[ DSC$W_LENGTH ] = .CLASS_S_DESC[ DSC$W_LENGTH ] + 1;
END;

++
--
Put a '.' on the end just like CVTROUH
CH$WCHAR(%C'.' , CH$PTR( .CLASS_S_DESC[ DSC$A_POINTER ] + ! A004
                          .CLASS_S_DESC[ DSC$W_LENGTH ])); ! A004

STATUS = SSS_NORMAL;                                         ! A004
END;                                                         ! A004

[K_DEC NBDS]:
BEGIN
  ! Don't support binary scale factor on packed.
  IF .BIN_SCALE NEQ 0
  THEN
    SDBG_ERROR('DBGCVTDX\DBG$CVT_DX_DX binary scale factor on packed not support

  NO DIGITS = .SRC_INFO [S_LEN];
  CVTIPS (NO DIGITS, INTMED_DATA, NO DIGITS, TEMP_BUF2);
  CLASS_S_DESC [DSC$W_LENGTH] = .NO_DIGITS + 1;
  OTSSCVT_H (CLASS_S_DESC, TEMP_BUF1, 0, 0,
  (K_IGN BLKS OR K_ENB UNDERFLOW OR K_IGN TABS ));
  STATUS = FOR$CVT_H_TF (TEMP_BUF1, CLASS_S_DESC, .DIGITS_IN_FRACT, .SCALE);
END;

TES:
BUF_OFFSET = CH$FIND_NOT_CH (K_TEMP_BUF_LENGTH, TEMP_BUF2, %C' ') - TEMP_BUF2;
NEXT_BLANK = CH$FIND_CH (K_TEMP_BUF_LENGTH - .BUF_OFFSET, TEMP_BUF2 + .BUF_OFFSET, %C' ');
IF .NEXT_BLANK EQL 0
THEN
  FINAL_LEN = K_TEMP_BUF_LENGTH - .BUF_OFFSET
ELSE
  FINAL_LEN = .NEXT_BLANK - .BUF_OFFSET - TEMP_BUF2;
IF .DIGITS_IN_FRACT EQL 0
THEN
  FINAL_LEN = .FINAL_LEN - 1;

IF NOT .STATUS
THEN
```

2889 3007 5  
2890 3008 5  
2891 3009 5  
2892 3010 5  
2893 3011 5  
2894 3012 5  
2895 3013 5  
2896 3014 5  
2897 3015 5  
2898 3016 5  
2899 3017 5  
2900 3018 5  
2901 3019 5  
2902 3020 5  
2903 3021 5  
2904 3022 4  
2905 3023 4  
2906 3024 4  
2907 3025 4  
2908 3026 4  
2909 3027 4  
2910 3028 5  
2911 3029 5  
2912 3030 5  
2913 3031 5  
2914 3032 5  
2915 3033 5  
2916 3034 5  
2917 3035 5  
2918 3036 5  
2919 3037 4  
2920 3038 4  
2921 3039 4  
2922 3040 5  
2923 3041 5  
2924 3042 5  
2925 3043 5  
2926 3044 5  
2927 3045 5  
2928 3046 5  
2929 3047 5  
2930 3048 5  
2931 3049 4  
2932 3050 4  
2933 3051 4  
2934 3052 5  
2935 3053 5  
2936 3054 5  
2937 3055 5  
2938 3056 4  
2939 3057 4  
2940 3058 3  
2941 3059 3  
2942 3060 3  
2943 3061 3  
2944 3062 3  
2945 3063 3

```

BEGIN
CLASS S_DESC [DSC$W_LENGTH] = K_TEMP_BUF_LENGTH;
IF .CVT_PATH EQL K_DEC_NBDS
THEN
  DIGITS_IN_FRACT = 31
ELSE
  IF .DST_INFO [D_LEN] - 9 LEQ 0
  THEN
    DIGITS_IN_FRACT = 33
  ELSE
    DIGITS_IN_FRACT = MIN (33, .DST_INFO [D_LEN] - 9);
STATUS = FOR$CVT RTE (TEMP_BUF1, CLASS_S_DESC, .DIGITS_IN_FRACT, .SCALE, 0, 4);
IF NOT .STATUS THEN $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: error in h-to-te conversio
BUF_OFFSET = CH$FIND NOT-CH (K_TEMP_BUF_LENGTH, TEMP_BUF2, %C' ') - TEMP_BUF2;
FINAL_LEN = K_TEMP_BUF_LENGTH - .BUF_OFFSET;
END;

OUTPUT_STR_LEN = .FINAL_LEN;
SELECTONE .DESTINATION[DSC$B_DTYPE] OF
SET
[DSC$K_DTYPE_AC]:
  BEGIN
  MAP
    OUTPUT: REF VECTOR[, BYTE];
    CLASS_S_DESC[DSC$W_LENGTH] = .FINAL_LEN;
    CLASS_S_DESC[DSC$A_POINTER] = OUTPUT[1];
    STATUS = LIB$SCOPY_R DX6 (.FINAL_LEN, TEMP_BUF2 + .BUF_OFFSET, CLASS_S_DESC);
    IF .STATUS EQL LIB$STRTRU THEN SIGNAL (DBG$_ISTRTRU, T, .DBG$GL_OPCODE_NAME);
    IF NOT .STATUS THEN SIGNAL (.STATUS);
    OUTPUT[0] = .FINAL_LEN;
  END;

[DSC$K_DTYPE_AZ]:
  BEGIN
  MAP
    OUTPUT: REF VECTOR[, BYTE];
    CLASS_S_DESC[DSC$W_LENGTH] = .FINAL_LEN;
    CLASS_S_DESC[DSC$A_POINTER] = OUTPUT[0];
    STATUS = LIB$SCOPY_R DX6 (.FINAL_LEN, TEMP_BUF2 + .BUF_OFFSET, CLASS_S_DESC);
    IF .STATUS EQL LIB$STRTRU THEN SIGNAL (DBG$_ISTRTRU, T, .DBG$GL_OPCODE_NAME);
    IF NOT .STATUS THEN SIGNAL (.STATUS);
    OUTPUT[.FINAL_LEN + 1] = 0;
  END;

[OTHERWISE]:
  BEGIN
    STATUS = LIB$SCOPY_R DX6 (.FINAL_LEN, TEMP_BUF2 + .BUF_OFFSET, .DESTINATION);
    IF .STATUS EQL LIB$STRTRU THEN SIGNAL (DBG$_ISTRTRU, T, .DBG$GL_OPCODE_NAME);
    IF NOT .STATUS THEN SIGNAL (.STATUS);
  END;
TES;
END;

[OTHERWISE]:
SELECTONE .CVT_PATH OF
SET
[K_SMLINT_NBDS]:

```

DBGCVTDX  
V04-000

C 8  
15-Sep-1984 23:57:30 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:16:44 [DEBUG.SRC]DBGCVTDX.B32;1

Page 70  
(16)

: 2946	3064	3
: 2947	3065	3
: 2948	3066	3
: 2949	3067	3
: 2950	3068	3
: 2951	3069	3
: 2952	3070	3
: 2953	3071	3

TES;

\$DBG\_ERROR ('DBGCVTDX\DBG\$CVT\_DX\_DX: smlint\_nbds');  
[K\_LRGINT\_NBDS]:  
\$DBG\_ERROR ('DBGCVTDX\DBG\$CVT\_DX\_DX: lrgint\_nbds');  
[K\_DEC\_NBDS]:  
\$DBG\_ERROR ('DBGCVTDX\DBG\$CVT\_DX\_DX: dec\_nbds');  
TES;

!For SMLINT\_NBDS, LRGINT\_NBDS, DEC\_NBDS

```

: 2955 3072 3
: 2956 3073 3
: 2957 3074 4
: 2958 3075 4
: 2959 3076 4
: 2960 3077 4
: 2961 3078 4
: 2962 3079 4
: 2963 3080 4
: 2964 3081 4
: 2965 3082 4
: 2966 3083 4
: 2967 3084 4
: 2968 3085 4
: 2969 3086 5
: 2970 3087 5
: 2971 3088 5
: 2972 3089 4
: 2973 3090 4
: 2974 3091 4
: 2975 3092 5
: 2976 3093 5
: 2977 3094 5
: 2978 3095 4
: 2979 3096 4
: 2980 3097 4
: 2981 3098 5
: 2982 3099 5
: 2983 3100 4
: 2984 3101 4
: 2985 3102 4
: 2986 3103 5
: 2987 3104 5
: 2988 3105 4
: 2989 3106 4
: 2990 3107 4
: 2991 3108 4
: 2992 3109 4
: 2993 3110 4
: 2994 3111 5
: 2995 3112 5
: 2996 3113 5
: 2997 3114 5
: 2998 3115 5
: 2999 3116 5
: 3000 3117 6
: 3001 3118 6
: 3002 3119 5
: 3003 3120 4
: 3004 3121 4
: 3005 3122 4
: 3006 3123 4
: 3007 3124 4
: 3008 3125 3

```

```

[K_SMLFLTCMPLX_SMLINT]:
  BEGIN
  M_SCALE_D D;
  IF .CVT_ROUND_FLAG
  THEN
    CVTRDL (INTMED_DATA, TEMP_BUF1)
  ELSE
    CVTDL (INTMED_DATA, TEMP_BUF1);
  CASE .DESTINATION [DSC$B_DTYPE] FROM DSC$K_DTYPE_V TO DSC$K_DTYPE_SVU OF
    SET
      [DSC$K_DTYPE_BU]:
        BEGIN
        IF .TEMP_BUF1 [LONG_1] GTRU K_LRGST_BU THEN SIGNAL (DBG$IINTOVF, 1, .DBG$GL_OPCODE_NAME
        OUTPUT [BYTE_1] = .TEMP_BUF1 [BYTE_1];
        END;
      [DSC$K_DTYPE_WU]:
        BEGIN
        IF .TEMP_BUF1 [LONG_1] GTRU K_LRGST_WU THEN SIGNAL (DBG$IINTOVF, 1, .DBG$GL_OPCODE_NAME
        OUTPUT [WORD_1] = .TEMP_BUF1 [WORD_1];
        END;
      [DSC$K_DTYPE_B]:
        BEGIN
        CVTLB (TEMP_BUF1, .OUTPUT);
        END;
      [DSC$K_DTYPE_W]:
        BEGIN
        CVTLW (TEMP_BUF1, .OUTPUT);
        END;
      [DSC$K_DTYPE_L]:
        OUTPUT [LONG_1] = .TEMP_BUF1 [S_LONG_1];
      [DSC$K_DTYPE_V, DSC$K_DTYPE_SV, DSC$K_DTYPE_VU, DSC$K_DTYPE_SVU, DSC$K_DTYPE_TF]:
        BEGIN
        MAP
          OUTPUT: REF BITVECTOR[K_OUTPUT_BUFFER_LENGTH * 8],
          INTMED_DATA: BITVECTOR[R_INTMED_DATA_LENGTH * 8];
          INCR I FROM 0 TO .DST_INFO[D_LEN] - 1 DO
            BEGIN
              OUTPUT[I] = .INTMED_DATA[I];
            END;
        END;
      [INRANGE, OUTRANGE]:
        $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: smlfltcmlx smlint');
      TES:
        !For SMLFLTCMPLX_SMLINT
    END;
  END;

```

```

: 3010
: 3011
: 3012
: 3013
: 3014
: 3015
: 3016
: 3017
: 3018
: 3019
: 3020
: 3021
: 3022
: 3023
: 3024
: 3025
: 3026
: 3027
: 3028
: 3029
: 3030
: 3031
: 3032
: 3033
: 3034
: 3035
: 3036
: 3037
: 3038
: 3039
: 3040
: 3041
: 3042

```

```

3126 3
3127 3
3128 4
3129 4
3130 4
3131 4
3132 4
3133 4
3134 5
3135 5
3136 5
3137 5
3138 5
3139 5
3140 5
3141 5
3142 5
3143 5
3144 4
3145 4
3146 4
3147 4
3148 4
3149 4
3150 5
3151 5
3152 5
3153 4
3154 4
3155 4
3156 4
3157 4
3158 3

```

```

[K_SMLFLTCMPLX_LRGINT]:
  BEGIN
  M_SCALE D D:
  CASE .DESTINATION [DSC$B_DTYPE] FROM DSC$K_DTYPE_LU TO DSC$K_DTYPE_O OF
    SET
      [DSC$K_DTYPE_LU]:
        BEGIN
        IF CMPD (INTMED_DATA, .LRGST_D_LU) GTR 0 THEN SIGNAL (DBG$IINTOVF, 1, .DBG$GL_OPCODE_NA
        BICPSW (%REF (K_SET_ARITHMETIC_TRAP));
        IF .CVT_ROUND_F[AG
        THEN
          CVTRDL (INTMED_DATA, .OUTPUT)
        ELSE
          CVTDL (INTMED_DATA, .OUTPUT);

        BISPSW (%REF (K_SET_ARITHMETIC_TRAP));
        END;
      [DSC$K_DTYPE_Q, DSC$K_DTYPE_QU]:
        CVTRDQ (INTMED_DATA, .OUTPUT);
      [DSC$K_DTYPE_O]:
        BEGIN
        CVTDH (INTMED_DATA, TEMP_BUF1);
        CVTRHO (TEMP_BUF1, .OUTPUT);
        END;
      [INRANGE, OUTRANGE]:
        $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: smlfltcmplx_lrgint');
        TES;
        !For SMLFLTCMPLX_LRGINT
  END;

```

```
3044 3159 3
3045 3160 3
3046 3161 3
3047 3162 3
3048 3163 3
3049 3164 3
3050 3165 4
3051 3166 4
3052 3167 4
3053 3168 4
3054 3169 5
3055 3170 5
3056 3171 5
3057 3172 5
3058 3173 5
3059 3174 5
3060 3175 5
3061 3176 5
3062 3177 5
3063 3178 5
3064 3179 4
3065 3180 4
3066 3181 4
3067 3182 4
3068 3183 4
3069 3184 4
3070 3185 4
3071 3186 4
3072 3187 4
3073 3188 4
3074 3189 4
3075 3190 4
3076 3191 4
3077 3192 4
3078 3193 5
3079 3194 5
3080 3195 5
3081 3196 5
3082 3197 5
3083 3198 5
3084 3199 5
3085 3200 5
3086 3201 5
3087 3202 4
3088 3203 4
3089 3204 4
3090 3205 5
3091 3206 5
3092 3207 5
3093 3208 5
3094 3209 5
3095 3210 5
3096 3211 5
3097 3212 5
3098 3213 5
3099 3214 4
3100 3215 4

[K_SMLFLT(CMPLX_NBDS):
  SELECTONE .DESTINATION [DSC$B_DTYPE] OF
    SET
      [DSC$K_DTYPE_BU, DSC$K_DTYPE_T, DSC$K_DTYPE_VT, DSC$K_DTYPE_AC, DSC$K_DTYPE_AZ]:
        BEGIN
          CLASS_S_DESC [DSC$W_LENGTH] = K_TEMP_BUF_LENGTH;
          CLASS_S_DESC [DSC$A_POINTER] = TEMP_BUF2;
          DIGITS_IN_FRACT =
            BEGIN
              CASE .SOURCE [DSC$B_DTYPE] FROM DSC$K_DTYPE_F TO DSC$K_DTYPE_D OF
                SET
                  [DSC$K_DTYPE_F]:
                    7;
                  [DSC$K_DTYPE_D]:
                    16;
                TES
              END;
            IF .DST_INFO [D_LEN] - 7 GTR 0
            THEN
              DIGITS_IN_FRACT = MIN (.DIGITS_IN_FRACT,
                .DST_INFO [D_LEN] - 7);
            STATUS = FOR$CVT D TE (INTMED DATA, CLASS_S_DESC, .DIGITS_IN_FRACT, .SCALE, 0);
            IF NOT .STATUS THEN $DBG_ERROR ('DBGCVTDX\DBG$CVT DX DX: error in d-to-te conversion');
            BUF_OFFSET = CH$FIND NOT_CH (K_TEMP_BUF_LENGTH, TEMP_BUF2, %C' ') - TEMP_BUF2;
            FINAL_LEN = K_TEMP_BUF_LENGTH - .BUF_OFFSET;
            OUTPUT_STR_LEN = .FINAL_LEN;
          SELECTONE .DESTINATION[DSC$B_DTYPE] OF
            SET
              [DSC$K_DTYPE_AC]:
                BEGIN
                  MAP
                    OUTPUT: REF VECTOR[, BYTE];
                    CLASS_S_DESC[DSC$W_LENGTH] = .FINAL_LEN;
                    CLASS_S_DESC[DSC$A_POINTER] = OUTPUT[1];
                    STATUS = LIB$SCOPY-R DX6 (.FINAL_LEN, TEMP_BUF2 + .BUF_OFFSET, CLASS_S_DESC);
                    IF .STATUS EQL LIB$_STRTRU THEN SIGNAL (DBG$_ISTRTRU, T, .DBG$GL_OPCODE_NAME);
                    IF NOT .STATUS THEN SIGNAL (.STATUS);
                    OUTPUT[0] = .FINAL_LEN;
                  END;
              [DSC$K_DTYPE_AZ]:
                BEGIN
                  MAP
                    OUTPUT: REF VECTOR[, BYTE];
                    CLASS_S_DESC[DSC$W_LENGTH] = .FINAL_LEN;
                    CLASS_S_DESC[DSC$A_POINTER] = OUTPUT[0];
                    STATUS = LIB$SCOPY-R DX6 (.FINAL_LEN, TEMP_BUF2 + .BUF_OFFSET, CLASS_S_DESC);
                    IF .STATUS EQL LIB$_STRTRU THEN SIGNAL (DBG$_ISTRTRU, T, .DBG$GL_OPCODE_NAME);
                    IF NOT .STATUS THEN SIGNAL (.STATUS);
                    OUTPUT[.FINAL_LEN + 1] = 0;
                  END;
```

DBGCVTDX  
V04-000

6 8  
15-Sep-1984 23:57:30 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:16:44 [DEBUG.SRC]DBGCVTDX.B32;1

Page 74  
(19)

:	3101	3216	4
:	3102	3217	5
:	3103	3218	5
:	3104	3219	5
:	3105	3220	5
:	3106	3221	4
:	3107	3222	4
:	3108	3223	3
:	3109	3224	3
:	3110	3225	3
:	3111	3226	3
:	3112	3227	3

```
[OTHERWISE]:
  BEGIN
    STATUS = LIB$COPY R DX6 (.FINAL_LEN, TEMP_BUF2 + .BUF_OFFSET, .DESTINATION);
    IF .STATUS EQL LIB$_STRTRU THEN SIGNAL (DBG$_ISTRTRU, T, .DBG$GL_OPCODE_NAME);
    IF NOT .STATUS THEN SIGNAL (.STATUS);
  END;
  TES;
END;

[OTHERWISE]:
  $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: smlfltcmplx_nbds');
  TES;
  !For SMLFLTCMPLX_NBDS
```



3114	3228	3
3115	3229	3
3116	3230	4
3117	3231	4
3118	3232	4
3119	3233	4
3120	3234	5
3121	3235	4
3122	3236	4
3123	3237	4
3124	3238	4
3125	3239	4
3126	3240	4
3127	3241	4
3128	3242	4
3129	3243	4
3130	3244	4
3131	3245	4
3132	3246	5
3133	3247	5
3134	3248	5
3135	3249	4
3136	3250	4
3137	3251	4
3138	3252	5
3139	3253	5
3140	3254	5
3141	3255	4
3142	3256	4
3143	3257	4
3144	3258	5
3145	3259	5
3146	3260	4
3147	3261	4
3148	3262	4
3149	3263	5
3150	3264	5
3151	3265	4
3152	3266	4
3153	3267	4
3154	3268	4
3155	3269	4
3156	3270	4
3157	3271	5
3158	3272	5
3159	3273	5
3160	3274	5
3161	3275	5
3162	3276	5
3163	3277	6
3164	3278	6
3165	3279	5
3166	3280	4
3167	3281	4
3168	3282	4
3169	3283	4
3170	3284	4

```

[K_LRGFLTCMPLX_SMLINT]:
BEGIN
  IF .SOURCE[DSC$B_DTYPE] EQL DSC$K_DTYPE_G OR
    .SOURCE[DSC$B_DTYPE] EQL DSC$K_DTYPE_GC
  THEN
    M_SCALE_G_H
  ELSE
    M_SCALE_H_H;
  IF .CVT_ROUND_FLAG
  THEN
    CVTRHL (INTMED_DATA, TEMP_BUF1)
  ELSE
    CVTHL (INTMED_DATA, TEMP_BUF1);
  CASE .DESTINATION [DSC$B_DTYPE] FROM DSC$K_DTYPE_V TO DSC$K_DTYPE_SVU OF
    SET
      [DSC$K_DTYPE_BU]:
        BEGIN
          IF .TEMP_BUF1 [LONG_1] GTRU K_LRGST_BU THEN SIGNAL (DBG$_IINTOVF, 1, .DBG$GL_OPCODE_NAME
            OUTPUT [BYTE_1] = .TEMP_BUF1 [BYTE_1];
          END;
        [DSC$K_DTYPE_WU]:
          BEGIN
            IF .TEMP_BUF1 [LONG_1] GTRU K_LRGST_WU THEN SIGNAL (DBG$_IINTOVF, 1, .DBG$GL_OPCODE_NAME
              OUTPUT [WORD_1] = .TEMP_BUF1 [WORD_1];
            END;
          [DSC$K_DTYPE_B]:
            BEGIN
              CVTLB (TEMP_BUF1, .OUTPUT);
            END;
          [DSC$K_DTYPE_W]:
            BEGIN
              CVTLW (TEMP_BUF1, .OUTPUT);
            END;
          [DSC$K_DTYPE_L]:
            OUTPUT [LONG_1] = .TEMP_BUF1 [S_LONG_1];
          [DSC$K_DTYPE_V, DSC$K_DTYPE_SV, DSC$K_DTYPE_VU, DSC$K_DTYPE_SVU, DSC$K_DTYPE_TF]:
            BEGIN
              MAP
                OUTPUT: REF BITVECTOR[K OUTPUT BUFFER LENGTH * 8],
                INTMED_DATA: BITVECTOR[K_INTMED_DATA_LENGTH * 8];
              INCR I FROM 0 TO .DST_INFO[LEN] - 1 DO
                BEGIN
                  OUTPUT[I] = .INTMED_DATA[I];
                END;
            END;
          [INRANGE, OUTRANGE]:
            $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: lrgfltcmplx smlint');
          TES;
          !For LRGFLTCMPLX_SMLINT

```

DBGCVTDX  
V04-000

; 3171

3285 3

END;

I 8  
15-Sep-1984 23:57:30  
14-Sep-1984 12:16:44

VAX-11 Bliss-32 V4.0-742  
[DEBUG.SRC]DBGCVTDX.B32;1

Page 76  
(20)

```

: 3173
: 3174
: 3175
: 3176
: 3177
: 3178
: 3179
: 3180
: 3181
: 3182
: 3183
: 3184
: 3185
: 3186
: 3187
: 3188
: 3189
: 3190
: 3191
: 3192
: 3193
: 3194
: 3195
: 3196
: 3197
: 3198
: 3199
: 3200
: 3201
: 3202
: 3203
: 3204
: 3205
: 3206

```

```

3286 3
3287 3
3288 4
3289 4
3290 4
3291 4
3292 5
3293 4
3294 4
3295 4
3296 4
3297 4
3298 4
3299 5
3300 5
3301 5
3302 5
3303 5
3304 5
3305 5
3306 5
3307 5
3308 4
3309 4
3310 4
3311 4
3312 4
3313 4
3314 4
3315 4
3316 4
3317 4
3318 4
3319 3

```

```

[K_LRGFLTCMPLX_LRGINT]:
BEGIN
IF .SOURCE[DSC$B_DTYPE] EQL DSC$K_DTYPE_G OR
.SOURCE[DSC$B_DTYPE] EQL DSC$K_DTYPE_GC
THEN
M_SCALE_G_H
ELSE
M_SCALE_H_H;
CASE .DESTINATION [DSC$B_DTYPE] FROM DSC$K_DTYPE_LU TO DSC$K_DTYPE_O OF
SET
[DSC$K_DTYPE_LU]:
BEGIN
IF CMPLX (INTMED_DATA, .LRGST_H_LU) GTR 0 THEN SIGNAL (DBG$_IINTOVF, 1, .DBG$GL_OPCODE_NA
BICPSW (%REF (K_SET_ARITHMETIC_TRAP));
IF .CVT_ROUND_FLAG
THEN
CVTRHL (INTMED_DATA, .OUTPUT)
ELSE
CVTHL (INTMED_DATA, .OUTPUT);
BISPSW (%REF (K_SET_ARITHMETIC_TRAP));
END;
[DSC$K_DTYPE_Q, DSC$K_DTYPE_QU]:
CVTRHQ (INTMED_DATA, .OUTPUT);
[DSC$K_DTYPE_O]:
CVTRHO (INTMED_DATA, .OUTPUT);
[INRANGE, OVRANGE]:
$DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: lrgfltcmplx lrgint');
TES;
!For LRGFLTCMPLX_LRGINT
END;

```

```

: 3208
: 3209
: 3210
: 3211
: 3212
: 3213
: 3214
: 3215
: 3216
: 3217
: 3218
: 3219
: 3220
: 3221
: 3222
: 3223
: 3224
: 3225
: 3226
: 3227
: 3228
: 3229
: 3230
: 3231
: 3232
: 3233
: 3234
: 3235
: 3236
: 3237
: 3238
: 3239
: 3240
: 3241
: 3242
: 3243
: 3244
: 3245
: 3246

```

```

3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358

```

```

[K_LRGFLTCMPLX_SMLFLTCMPLX]:
BEGIN
  IF .SOURCE[DSC$B_DTYPE] EQL DSC$K_DTYPE_G OR
    .SOURCE[DSC$B_DTYPE] EQL DSC$K_DTYPE_GC
  THEN
    M_SCALE_G_H
  ELSE
    M_SCALE_H_H;
  CASE .DESTINATION [DSC$B_DTYPE] FROM DSC$K_DTYPE_F TO DSC$K_DTYPE_D OF
    SET
      [DSC$K_DTYPE_F]:
        CVTHF (INTMED_DATA, .OUTPUT);
      [DSC$K_DTYPE_D]:
        CVTHD (INTMED_DATA, .OUTPUT);
      [INRANGE, OUTRANGE]:
        CASE .DESTINATION[DSC$B_DTYPE] FROM DSC$K_DTYPE_FC TO DSC$K_DTYPE_DC OF
          SET
            [DSC$K_DTYPE_FC]:
              BEGIN
                CVTHF (INTMED_DATA, .OUTPUT);
                CVTHF (INTMED_DATA+16, .OUTPUT+4);
              END;
            [DSC$K_DTYPE_DC]:
              BEGIN
                CVTHD (INTMED_DATA, .OUTPUT);
                CVTHD (INTMED_DATA+16, .OUTPUT+8);
              END;
          [INRANGE, OUTRANGE]:
            $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: lrgfltcmlx_smlfltcmlx');
          TES;
        !For LRGFLTCMPLX_SMLFLTCMPLX
      TES;
    END;
  END;

```

```

3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304

```

```

[K_LRGFLTCMPLEX_NBDS]:
  SELECTONE .DESTINATION [DSC$B_DTYPE] OF
  SET
    [DSC$K_DTYPE_BU, DSC$K_DTYPE_T, DSC$K_DTYPE_VT, DSC$K_DTYPE_AC, DSC$K_DTYPE_AZ]:
    BEGIN
      LOCAL
        DIGITS_IN_EXP,
        NOT_DIGITS;

      CLASS_S_DESC [DSC$ LENGTH] = K TEMP BUF LENGTH;
      CLASS_S_DESC [DSC$ POINTER] = TEMP BUF2;
      CASE .SOURCE [DSC$B_DTYPE] FROM DSC$K_DTYPE_G TO DSC$K_DTYPE_H OF
        SET
          [DSC$K_DTYPE_G]:
          BEGIN
            DIGITS_IN_FRACT = 15;
            DIGITS_IN_EXP = 3;
            NOT_DIGITS = 7;
            IF .DST_INFO [D_LEN] - .NOT_DIGITS GTR 0
            THEN
              DIGITS_IN_FRACT = MIN (.DIGITS_IN_FRACT, .DST_INFO [D_LEN] - .NOT_DIGITS);
              STATUS = FOR$CVT G TE (INTMED DATA, CLASS_S_DESC, .DIGITS_IN_FRACT, .SCALE, 0, .
              IF NOT .STATUS THEN $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: error in g-to-te conve
              END;
          [DSC$K_DTYPE_H]:
          BEGIN
            DIGITS_IN_FRACT = 33;
            DIGITS_IN_EXP = 4;
            NOT_DIGITS = 8;
            IF .DST_INFO [D_LEN] - .NOT_DIGITS GTR 0
            THEN
              DIGITS_IN_FRACT = MIN (.DIGITS_IN_FRACT, .DST_INFO [D_LEN] - .NOT_DIGITS);
              STATUS = FOR$CVT H TE (INTMED DATA, CLASS_S_DESC, .DIGITS_IN_FRACT, .SCALE, 0, .
              IF NOT .STATUS THEN $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: error in h-to-te conve
              END;
          TES;

      BUF_OFFSET = CH$FIND NOT CH (K TEMP BUF LENGTH, TEMP_BUF2, %C' ') - TEMP_BUF2;
      FINAL_LEN = K TEMP BUF LENGTH - .BUF_OFFSET;
      OUTPUT_STR_LEN = .FINAL_LEN;

      SELECTONE .DESTINATION[DSC$B_DTYPE] OF
      SET
        [DSC$K_DTYPE_AC]:
        BEGIN
          MAP
            OUTPUT: REF VECTOR[, BYTE];
            CLASS_S_DESC[DSC$W_LENGTH] = .FINAL_LEN;
            CLASS_S_DESC[DSC$A_POINTER] = OUTPUT[1];
            STATUS = LIB$COPY R DX6 (.FINAL_LEN, TEMP_BUF2 + .BUF_OFFSET, CLASS_S_DESC);
            IF .STATUS EQL LIB$STRTRU THEN SIGNAL (DBG$ISTRTRU, T, .DBG$GL_OPCODE_NAME);
            IF NOT .STATUS THEN SIGNAL (.STATUS);
            OUTPUT[0] = .FINAL_LEN;
          END;

```

DBGCVTDX  
V04-000

M 8  
15-Sep-1984 23:57:30  
14-Sep-1984 12:16:44

VAX-11 Bliss-32 V4.0-742  
[DEBUG.SRC]DBGCVTDX.B32;1

Page 2)  
(23)

```

: 3305      3416  4
: 3306      3417  4
: 3307      3418  5
: 3308      3419  5
: 3309      3420  5
: 3310      3421  5
: 3311      3422  5
: 3312      3423  5
: 3313      3424  5
: 3314      3425  5
: 3315      3426  5
: 3316      3427  4
: 3317      3428  4
: 3318      3429  4
: 3319      3430  5
: 3320      3431  5
: 3321      3432  5
: 3322      3433  5
: 3323      3434  4
: 3324      3435  4
: 3325      3436  3
: 3326      3437  3
: 3327      3438  3
: 3328      3439  3
: 3329      3440  3

```

```

[DSC$K_DTYPE_AZ]:
  BEGIN
  MAP
    OUTPUT: REF VECTOR[, BYTE];
    CLASS_S_DESC[DSC$W_LENGTH] = .FINAL_LEN;
    CLASS_S_DESC[DSC$A_POINTER] = OUTPUT[0];
    STATUS = LIB$SCOPY R DX6 (.FINAL_LEN, TEMP BUF2 + .BUF_OFFSET, CLASS_S_DESC);
    IF .STATUS EQL LIB$STRTRU THEN SIGNAL (DBG$_ISTRTRU, T, .DBG$GL_OPCODE_NAME);
    IF NOT .STATUS THEN SIGNAL (.STATUS);
    OUTPUT[.FINAL_LEN + 1] = 0;
  END;

[OTHERWISE]:
  BEGIN
    STATUS = LIB$SCOPY R DX6 (.FINAL_LEN, TEMP BUF2 + .BUF_OFFSET, .DESTINATION);
    IF .STATUS EQL LIB$STRTRU THEN SIGNAL (DBG$_ISTRTRU, T, .DBG$GL_OPCODE_NAME);
    IF NOT .STATUS THEN SIGNAL (.STATUS);
  END;
  TES;
END;

[OTHERWISE]:
  $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: lrgfltcmplx_nbds');
TES;
!For LRGFLTCMPLX_NBDS

```

```

3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387

```

```

[K_DEC_SMLINT]:
BEGIN
  IF .DESTINATION [DSC$V_FL_BINSCALE]
  THEN
    BEGIN
      This is a HACK for scaled binary. The Idea is to run the
      scaled packed decimal up to H_Float and then back down to
      the particular dtype below. The algorithm is as follows:

      The destination is a binary scale type so the conversion is
      done by hand.
      1) Get the sign.
      2) Get the scale of the H_Float.
      3) Check if an overflow will occur. An underflow is
         acceptable and will be truncated automatically.
      4) Move the most significant H_Float fractional bits
         into the temporary destination.
         (Note: this includes the redundant most significant
         fraction bit.
      5) Alter the destination to the correct scale.
      6) This is an absolute value so correct for the sign.
      7) Move the result into the final destination.

      ***** HACK - BAB Dec. 1983 *****

    M SCALE_P_H;
    END
  ELSE
    BEGIN
      M SCALE_P_P;
      CVTPL (NO_DIGITS, INTMED_DATA, TEMP_BUF1);
      END;
  CASE .DESTINATION [DSC$B_DTYPE] FROM DSC$K_DTYPE_V TO DSC$K_DTYPE_SVU OF
  SET
    [DSC$K_DTYPE_BU]:
    BEGIN
      ! If the target is not a binary scale, then just move the
      ! converted value in.
      IF NOT .DESTINATION [DSC$V_FL_BINSCALE]
      THEN
        BEGIN
          IF .TEMP_BUF1 [LONG_1] GTRU K_LRGST_BU
          THEN
            SIGNAL (DBG$IINTOVF, 1, DBG$GL_OPCODE_NAME);
            OUTPUT [BYTE_1] = .TEMP_BUF1 [BYTE_1];
            END
          ELSE
            ! If the sign and the scale of the H_Float are zero,
            ! then the value is zero.

```

```

3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444

```

```

3498 5
3499 5
3500 5
3501 5
3502 6
3503 6
3504 6
3505 6
3506 6
3507 7
3508 6
3509 6
3510 6
3511 7
3512 7
3513 7
3514 7
3515 7
3516 8
3517 8
3518 8
3519 7
3520 7
3521 7
3522 6
3523 5
3524 4
3525 4
3526 4
3527 5
3528 5
3529 5
3530 5
3531 5
3532 5
3533 5
3534 6
3535 6
3536 6
3537 6
3538 6
3539 6
3540 5
3541 5
3542 5
3543 5
3544 5
3545 5
3546 5
3547 5
3548 5
3549 6
3550 6
3551 6
3552 6
3553 6
3554 7

```

```

IF .INTMED_DATA[WORD_1] EQL 0
THEN
  OUTPUT[BYTE_1] = 0
ELSE
  BEGIN
    TEMP_BUF1 = 0;
    SIGN = .INTMED_DATA<15, 1, 0>;
    INTMED_DATA<15, 1, 0> = 0;
    FLOAT_SCALE = .INTMED_DATA[WORD_1] - 16384;
    IF .FLOAT_SCALE GTR (7 + .DESTINATION[DSC$B_SCALE])
    THEN
      SIGNAL(DBG$_INTOVF, 1, .DBG$GL_OPCODE_NAME)
    ELSE
      BEGIN
        TEMP_BUF1<6, 1, 0> = 1;
        TEMP_BUF1<0, 6, 0> = .INTMED_DATA<26, 6, 0>;
        FLOAT_SCALE = 7 + .DESTINATION[DSC$B_SCALE] - .FLOAT_SCALE;
        WHILE .FLOAT_SCALE GTR 0 DO
          BEGIN
            TEMP_BUF1[LONG_1] = .TEMP_BUF1[S_LONG_1] / 2;
            FLOAT_SCALE = .FLOAT_SCALE - 1;
          END;
        IF .SIGN THEN TEMP_BUF1 = 0 - .TEMP_BUF1;
        OUTPUT[BYTE_1] = .TEMP_BUF1[S_BYTE_1];
      END;
    END;
  END;
END;

[DSC$K_DTYPE_WU]:
BEGIN
  ! If the target is not a binary scale, then just move the
  ! converted value in.
  !
  IF NOT .DESTINATION[DSC$V_FL_BINSCALE]
  THEN
    BEGIN
      IF .TEMP_BUF1[LONG_1] GTRU K_LRGST_WU
      THEN
        SIGNAL(DBG$_INTOVF, 1, .DBG$GL_OPCODE_NAME);
        OUTPUT[WORD_1] = .TEMP_BUF1[WORD_1];
      END
    ELSE
      ! If the sign and the scale of the H_Float are zero,
      ! then the value is zero.
      !
      IF .INTMED_DATA[WORD_1] EQL 0
      THEN
        OUTPUT[WORD_1] = 0
      ELSE
        BEGIN
          TEMP_BUF1 = 0;
          SIGN = .INTMED_DATA<15, 1, 0>;
          INTMED_DATA<15, 1, 0> = 0;
          FLOAT_SCALE = .INTMED_DATA[WORD_1] - 16384;
          IF .FLOAT_SCALE GTR (15 + .DESTINATION[DSC$B_SCALE])

```



3445	3555	6	THEN
3446	3556	6	SIGNAL(DBG\$_IINTOVF, 1, .DBG\$_GL_OPCODE_NAME)
3447	3557	6	ELSE
3448	3558	7	BEGIN
3449	3559	7	TEMP_BUF1<14, 1, 0> = 1;
3450	3560	7	TEMP_BUF1<0, 14, 0> = .INTMED_DATA<18, 14, 0>;
3451	3561	7	FLOAT_SCALE = 15 + .DESTINATION[DS\$_B_SCALE] - .FLOAT_SCALE;
3452	3562	7	WHILE .FLOAT_SCALE GTR 0 DO
3453	3563	8	BEGIN
3454	3564	8	TEMP_BUF1[LONG_1] = .TEMP_BUF1[S_LONG_1] / 2;
3455	3565	8	FLOAT_SCALE = .FLOAT_SCALE - 1;
3456	3566	7	END;
3457	3567	7	IF .SIGN THEN TEMP_BUF1 = 0 - .TEMP_BUF1;
3458	3568	7	OUTPUT [WORD_1] = .TEMP_BUF1 [S_WORD_1];
3459	3569	6	END;
3460	3570	5	END;
3461	3571	4	END;
3462	3572	4	[DS\$_K_DTYPE_B]:
3463	3573	4	BEGIN
3464	3574	5	! If the target is not a binary scale, then just move the
3465	3575	5	! converted value in.
3466	3576	5	! If the target is not a binary scale, then just move the
3467	3577	5	! converted value in.
3468	3578	5	! If the target is not a binary scale, then just move the
3469	3579	5	! converted value in.
3470	3580	5	IF NOT .DESTINATION [DS\$_V_FL_BINSKALE]
3471	3581	5	THEN
3472	3582	5	CVTLB (TEMP_BUF1, .OUTPUT)
3473	3583	5	ELSE
3474	3584	5	! If the sign and the scale of the H_Float are zero,
3475	3585	5	! then the value is zero.
3476	3586	5	! If the sign and the scale of the H_Float are zero,
3477	3587	5	! then the value is zero.
3478	3588	5	IF .INTMED_DATA[WORD_1] EQL 0
3479	3589	5	THEN
3480	3590	5	OUTPUT[BYTE_1] = 0
3481	3591	6	ELSE
3482	3592	6	BEGIN
3483	3593	6	TEMP_BUF1 = 0;
3484	3594	6	SIGN = .INTMED_DATA<15, 1, 0>;
3485	3595	6	INTMED_DATA<15, 1, 0> = 0;
3486	3596	7	FLOAT_SCALE = .INTMED_DATA[WORD_1] - 16384;
3487	3597	6	IF .FLOAT_SCALE GTR (7 + .DESTINATION[DS\$_B_SCALE])
3488	3598	6	THEN
3489	3599	6	SIGNAL(DBG\$_IINTOVF, 1, .DBG\$_GL_OPCODE_NAME)
3490	3600	7	ELSE
3491	3601	7	BEGIN
3492	3602	7	TEMP_BUF1<6, 1, 0> = 1;
3493	3603	7	TEMP_BUF1<0, 6, 0> = .INTMED_DATA<26, 6, 0>;
3494	3604	7	FLOAT_SCALE = 7 + .DESTINATION[DS\$_B_SCALE] - .FLOAT_SCALE;
3495	3605	8	WHILE .FLOAT_SCALE GTR 0 DO
3496	3606	8	BEGIN
3497	3607	8	TEMP_BUF1[LONG_1] = .TEMP_BUF1[S_LONG_1] / 2;
3498	3608	7	FLOAT_SCALE = .FLOAT_SCALE - 1;
3499	3609	7	END;
3500	3610	7	IF .SIGN THEN TEMP_BUF1 = 0 - .TEMP_BUF1;
3501	3611	6	OUTPUT [BYTE_1] = .TEMP_BUF1 [S_BYTE_1];
			END;

```

3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558

```

```

END;
END;
[ DSC$K_DTYPE_W ]:
BEGIN
    ! If the target is not a binary scale, then just move the
    ! converted value in.
    IF NOT .DESTINATION [ DSC$V_FL_BINSCALE ]
    THEN
        CVTLW ( TEMP_BUF1, .OUTPUT )
    ELSE
        ! If the sign and the scale of the H_Float are zero,
        ! then the value is zero.
        IF .INTMED_DATA[WORD_1] EQL 0
        THEN
            OUTPUT[WORD_1] = 0
        ELSE
            BEGIN
                TEMP_BUF1 = 0;
                SIGN = .INTMED_DATA<15, 1, 0>;
                INTMED_DATA<15, 1, 0> = 0;
                FLOAT_SCALE = .INTMED_DATA[WORD_1] - 16384;
                IF .FLOAT_SCALE GTR ( 15 + .DESTINATION[ DSC$B_SCALE ] )
                THEN
                    SIGNAL( DBG$ _INTOVF, 1, .DBG$GL_OPCODE_NAME )
                ELSE
                    BEGIN
                        TEMP_BUF1<14, 1, 0> = 1;
                        TEMP_BUF1<0, 14, 0> = .INTMED_DATA<18, 14, 0>;
                        FLOAT_SCALE = 15 + .DESTINATION[ DSC$B_SCALE ] - .FLOAT_SCALE;
                        WHILE .FLOAT_SCALE GTR 0 DO
                            BEGIN
                                TEMP_BUF1[ LONG_1 ] = .TEMP_BUF1[ S_LONG_1 ] / 2;
                                FLOAT_SCALE = .FLOAT_SCALE - 1;
                            END;
                        IF .SIGN THEN TEMP_BUF1 = 0 - .TEMP_BUF1;
                        OUTPUT [WORD_1] = .TEMP_BUF1 [ S_WORD_1 ];
                    END;
                END;
            END;
        END;
    END;
[ DSC$K_DTYPE_L ]:
BEGIN
    ! If the target is not a binary scale, then just move the
    ! converted value in.
    IF NOT .DESTINATION [ DSC$V_FL_BINSCALE ]
    THEN
        OUTPUT [ LONG_1 ] = .TEMP_BUF1 [ S_LONG_1 ]
    ELSE
        ! If the sign and the scale of the H_Float are zero,

```

```

: 3559 3669 5
: 3560 3670 5
: 3561 3671 5
: 3562 3672 5
: 3563 3673 5
: 3564 3674 5
: 3565 3675 6
: 3566 3676 6
: 3567 3677 6
: 3568 3678 6
: 3569 3679 6
: 3570 3680 7
: 3571 3681 6
: 3572 3682 6
: 3573 3683 6
: 3574 3684 7
: 3575 3685 7
: 3576 3686 7
: 3577 3687 7
: 3578 3688 7
: 3579 3689 7
: 3580 3690 8
: 3581 3691 8
: 3582 3692 8
: 3583 3693 7
: 3584 3694 7
: 3585 3695 7
: 3586 3696 6
: 3587 3697 5
: 3588 3698 4
: 3589 3699 4
: 3590 3700 4
: 3591 3701 5
: 3592 3702 5
: 3593 3703 5
: 3594 3704 5
: 3595 3705 5
: 3596 3706 5
: 3597 3707 6
: 3598 3708 6
: 3599 3709 5
: 3600 3710 4
: 3601 3711 4
: 3602 3712 4
: 3603 3713 4
: 3604 3714 4
: 3605 3715 3

```

```

! then the value is zero.
IF .INTMED_DATA[WORD_1] EQL 0
THEN
  OUTPUT[LONG_1] = 0
ELSE
  BEGIN
    TEMP_BUF1 = 0;
    SIGN = .INTMED_DATA<15, 1, 0>;
    INTMED_DATA<15, 1, 0> = 0;
    FLOAT_SCALE = .INTMED_DATA[WORD_1] - 16384;
    IF .FLOAT_SCALE GTR (31 + .DESTINATION[DSC$B_SCALE])
    THEN
      SIGNAL(DBG$_INTOVF, 1, .DBG$GL_OPCODE_NAME)
    ELSE
      BEGIN
        TEMP_BUF1<30, 1, 0> = 1;
        TEMP_BUF1<14, 16, 0> = .INTMED_DATA<16, 16, 0>;
        TEMP_BUF1<0, 14, 0> = .(INTMED_DATA+4)<18, 14, 0>;
        FLOAT_SCALE = 31 + .DESTINATION[DSC$B_SCALE] - .FLOAT_SCALE;
        WHILE .FLOAT_SCALE GTR 0 DO
          BEGIN
            TEMP_BUF1[LONG_1] = .TEMP_BUF1[S_LONG_1] / 2;
            FLOAT_SCALE = .FLOAT_SCALE - 1;
          END;
        IF .SIGN THEN TEMP_BUF1 = 0 - .TEMP_BUF1;
        OUTPUT [LONG_1] = .TEMP_BUF1 [S_LONG_1];
      END;
    END;
  END;
END;

[DSC$K_DTYPE_V, DSC$K_DTYPE_SV, DSC$K_DTYPE_VU, DSC$K_DTYPE_SVU, DSC$K_DTYPE_VF]:
BEGIN
  MAP
    OUTPUT: REF BITVECTOR[K OUTPUT BUFFER LENGTH * 8],
    INTMED_DATA: BITVECTOR[R_INTMED_DATA_LENGTH * 8];

  INCR I FROM 0 TO .DST_INFO[D_LEN] - 1 DO
    BEGIN
      OUTPUT[I] = .INTMED_DATA[I];
    END;
  END;
END;

[INRANGE, OUTRANGE]:
$DBG_ERROR ('DBGCVTDX\DBG$CVT DX_DX: dec smlint');
TES;
!For DEC_SMLINT
END;

```

3607	3716	3
3608	3717	3
3609	3718	4
3610	3719	4
3611	3720	4
3612	3721	4
3613	3722	4
3614	3723	4
3615	3724	4
3616	3725	5
3617	3726	6
3618	3727	5
3619	3728	5
3620	3729	5
3621	3730	5
3622	3731	5
3623	3732	4
3624	3733	4
3625	3734	4
3626	3735	5
3627	3736	5
3628	3737	5
3629	3738	5
3630	3739	5
3631	3740	5
3632	3741	5
3633	3742	5
3634	3743	5
3635	3744	5
3636	3745	5
3637	3746	4
3638	3747	4
3639	3748	4
3640	3749	4
3641	3750	4
3642	3751	3

```

[K_DEC_LRGINT]:
  BEGIN
    M_SCALE_P_P;
    CASE .DESTINATION [DSC$B_DTYPE] FROM DSC$K_DTYPE_LU TO DSC$K_DTYPE_O OF
      SET
        [DSC$K_DTYPE_LU]:
          BEGIN
            IF (CMPP (NO_DIGITS, INTMED_DATA, %REF (K_PACK_LU_LEN), .LRGST_P_LU) GEQ 0)
              THEN
                SIGNAL (DBG$ IINTOVF, 1, .DBG$GL_OPCODE_NAME);
                BICPSW (%REF (K_SET_ARITHMETIC_TRAP));
                CVTPL (NO_DIGITS, INTMED_DATA, .OUTPUT);
                BISPSW (%REF (K_SET_ARITHMETIC_TRAP));
              END;
        [DSC$K_DTYPE_Q, DSC$K_DTYPE_QU, DSC$K_DTYPE_O]:
          BEGIN
            CVTPS (NO_DIGITS, INTMED_DATA, NO_DIGITS, TEMP_BUF2);
            CLASS_S_DESC [DSC$W_LENGTH] = .NO_DIGITS + 1;
            CLASS_S_DESC [DSC$A_POINTER] = TEMP_BUF2;
            OTSSCVT-T-H (CLASS_S_DESC, TEMP_BUF1);
            IF .DESTINATION[DSC$B_DTYPE] EQ[ DSC$K_DTYPE_Q OR
              .DESTINATION[DSC$B_DTYPE] EQL DSC$K_DTYPE_QU
            THEN
              CVTRHQ (TEMP_BUF1, .OUTPUT)
            ELSE
              CVTRHO (TEMP_BUF1, OUTPUT);
            END;
          [INRANGE, OUTRANGE]:
            $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX-DX: dec lrgint');
          TES;
          !For DEC_LRGINT
        END;
      END;
    END;
  
```

```
: 3644 3752 3
: 3645 3753 3
: 3646 3754 4
: 3647 3755 4
: 3648 3756 4
: 3649 3757 4
: 3650 3758 4
: 3651 3759 4
: 3652 3760 4
: 3653 3761 4
: 3654 3762 4
: 3655 3763 4
: 3656 3764 4
: 3657 3765 4
: 3658 3766 4
: 3659 3767 4
: 3660 3768 4
: 3661 3769 4
: 3662 3770 4
: 3663 3771 4
: 3664 3772 4
: 3665 3773 4
: 3666 3774 4
: 3667 3775 4
: 3668 3776 4
: 3669 3777 4
: 3670 3778 4
: 3671 3779 4
: 3672 3780 4
: 3673 3781 4
: 3674 3782 4
: 3675 3783 4
: 3676 3784 4
: 3677 3785 4
: 3678 3786 4
: 3679 3787 4
: 3680 3788 4
: 3681 3789 4
: 3682 3790 4
: 3683 3791 4
: 3684 3792 5
: 3685 3793 5
: 3686 3794 5
: 3687 3795 5
: 3688 3796 5
: 3689 3797 5
: 3690 3798 5
: 3691 3799 6
: 3692 3800 6
: 3693 3801 6
: 3694 3802 6
: 3695 3803 6
: 3696 3804 6
: 3697 3805 5
: 3698 3806 5
: 3699 3807 5
: 3700 3808 5
```

```
[K_NBDS_SMLINT]:
```

```
BEGIN
CLASS_S_DESC [DSC$W_LENGTH] = .SRC_INFO [S_LEN];
CLASS_S_DESC [DSC$A_POINTER] = .SRC_INFO [S_POINTER];
STATUS = OT$SCVT T_R (CLASS_S_DESC, TEMP_BUF1, 0, -.SCALE,
(K_IGN_BKLS OR K_ENB_UNDERFLOW OR K_IGN_TABS OR K_ENB_SCALE));
IF NOT .STATUS THEN SIGNAL (DBG$_INVNUMSTR, 1, .DBG$GL_OPCODE_NAME);
```

```
! This is a HACK for scaled binary. If the destination is Scaled
! Binary we will leave the value as a H_Float so that we can
! do the convert to Scaled Binary by hand. The algorithm follows:
```

```
! This is the algorithm for the code in the particular case below:
```

- 1) Get the sign.
- 2) Get the scale of the H\_Float.
- 3) Check if an overflow will occur. An underflow is acceptable and will be truncated automatically.
- 4) Move the most significant H\_Float fractional bits into the temporary destination.  
(Note: this includes the redundant most significant fraction bit.
- 5) Alter the destination to the correct scale.
- 6) This is an absolute value so correct for the sign.
- 7) Move the result into the final destination.

```
***** HACK - BAB Dec. 1983 *****
```

```
IF NOT .DESTINATION [DSC$V_FL_BINSKALE]
THEN
```

```
IF .CVT_ROUND_FLAG
```

```
THEN
```

```
CVTRHL (TEMP_BUF1, TEMP_BUF2)
```

```
ELSE
```

```
CVTHL (TEMP_BUF1, TEMP_BUF2);
```

```
CASE .DESTINATION [DSC$B_DTYPE] FROM DSC$K_DTYPE_V TO DSC$K_DTYPE_SVU OF
SET
```

```
[DSC$K_DTYPE_BU]:
```

```
BEGIN
```

```
! If the target is not a binary scale, then just move the
! converted value in.
```

```
IF NOT .DESTINATION [DSC$V_FL_BINSKALE]
```

```
THEN
```

```
BEGIN
```

```
IF .TEMP_BUF2 [LONG_1] GTRU K_LRGST_BU
```

```
THEN
```

```
SIGNAL (DBG$_IINTOVF, 1, .DBG$GL_OPCODE_NAME);
```

```
OUTPUT [BYTE_1] = .TEMP_BUF2 [BYTE_1];
```

```
END
```

```
ELSE
```

```
! If the sign and the scale of the H_Float are zero,
! then the value is zero.
```

```

3701 3809 5
3702 3810 5
3703 3811 5
3704 3812 5
3705 3813 5
3706 3814 6
3707 3815 6
3708 3816 6
3709 3817 6
3710 3818 6
3711 3819 7
3712 3820 6
3713 3821 6
3714 3822 6
3715 3823 7
3716 3824 7
3717 3825 7
3718 3826 7
3719 3827 7
3720 3828 8
3721 3829 8
3722 3830 8
3723 3831 7
3724 3832 7
3725 3833 7
3726 3834 6
3727 3835 5
3728 3836 4
3729 3837 4
3730 3838 4
3731 3839 5
3732 3840 5
3733 3841 5
3734 3842 5
3735 3843 5
3736 3844 5
3737 3845 5
3738 3846 6
3739 3847 6
3740 3848 6
3741 3849 6
3742 3850 6
3743 3851 6
3744 3852 5
3745 3853 5
3746 3854 5
3747 3855 5
3748 3856 5
3749 3857 5
3750 3858 5
3751 3859 5
3752 3860 5
3753 3861 6
3754 3862 6
3755 3863 6
3756 3864 6
3757 3865 6

```

```

!
IF .INTMED_DATA[WORD_1] EQL 0
THEN
    OUTPUT[BYTE_1] = 0
ELSE
    BEGIN
        TEMP_BUF1 = 0;
        SIGN = .INTMED_DATA<15, 1, 0>;
        INTMED_DATA<15, 1, 0> = 0;
        FLOAT_SCALE = .INTMED_DATA[WORD_1] - 16384;
        IF .FLOAT_SCALE GTR (7 + .DESTINATION[DSC$B_SCALE])
        THEN
            SIGNAL(DBG$IINTOVF, 1, .DBG$GL_OPCODE_NAME)
        ELSE
            BEGIN
                TEMP_BUF1<6, 1, 0> = 1;
                TEMP_BUF1<0, 6, 0> = .INTMED_DATA<26, 6, 0>;
                FLOAT_SCALE = 7 + .DESTINATION[DSC$B_SCALE] - .FLOAT_SCALE;
                WHILE .FLOAT_SCALE GTR 0 DO
                    BEGIN
                        TEMP_BUF1[LONG_1] = .TEMP_BUF1[S_LONG_1] / 2;
                        FLOAT_SCALE = .FLOAT_SCALE - 1;
                    END;
                IF .SIGN THEN TEMP_BUF1 = 0 - .TEMP_BUF1;
                OUTPUT[BYTE_1] = .TEMP_BUF1[S_BYTE_1];
            END;
        END;
    END;

[DSC$K_DTYPE_WU]:
BEGIN
    ! If the target is not a binary scale, then just move the
    ! converted value in.
    !
    IF NOT .DESTINATION[DSC$V_FL_BINSKALE]
    THEN
        BEGIN
            IF .TEMP_BUF2[LONG_1] GTRU K_LRGST_WU
            THEN
                SIGNAL(DBG$IINTOVF, 1, .DBG$GL_OPCODE_NAME);
                OUTPUT[WORD_1] = .TEMP_BUF2[WORD_1];
            END
        ELSE
            ! If the sign and the scale of the H_Float are zero,
            ! then the value is zero.
            !
            IF .INTMED_DATA[WORD_1] EQL 0
            THEN
                OUTPUT[WORD_1] = 0
            ELSE
                BEGIN
                    TEMP_BUF1 = 0;
                    SIGN = .INTMED_DATA<15, 1, 0>;
                    INTMED_DATA<15, 1, 0> = 0;
                    FLOAT_SCALE = .INTMED_DATA[WORD_1] - 16384;

```

```

: 3758
: 3759
: 3760
: 3761
: 3762
: 3763
: 3764
: 3765
: 3766
: 3767
: 3768
: 3769
: 3770
: 3771
: 3772
: 3773
: 3774
: 3775
: 3776
: 3777
: 3778
: 3779
: 3780
: 3781
: 3782
: 3783
: 3784
: 3785
: 3786
: 3787
: 3788
: 3789
: 3790
: 3791
: 3792
: 3793
: 3794
: 3795
: 3796
: 3797
: 3798
: 3799
: 3800
: 3801
: 3802
: 3803
: 3804
: 3805
: 3806
: 3807
: 3808
: 3809
: 3810
: 3811
: 3812
: 3813
: 3814

```

```

3866 7
3867 6
3868 6
3869 6
3870 7
3871 7
3872 7
3873 7
3874 7
3875 8
3876 8
3877 8
3878 7
3879 7
3880 7
3881 6
3882 5
3883 4
3884 4
3885 4
3886 5
3887 5
3888 5
3889 5
3890 5
3891 5
3892 5
3893 5
3894 5
3895 5
3896 5
3897 5
3898 5
3899 5
3900 5
3901 5
3902 5
3903 6
3904 6
3905 6
3906 6
3907 6
3908 7
3909 6
3910 6
3911 6
3912 7
3913 7
3914 7
3915 7
3916 7
3917 8
3918 8
3919 8
3920 7
3921 7
3922 7

```

```

IF .FLOAT_SCALE GTR (15 + .DESTINATION[DSC$B_SCALE])
THEN
    SIGNAL(DBG$_IINTOVF, 1, .DBG$GL_OPCODE_NAME)
ELSE
    BEGIN
        TEMP_BUF1<14, 1, 0> = 1;
        TEMP_BUF1<0, 14, 0> = .INTMED_DATA<18, 14, 0>;
        FLOAT_SCALE = 15 + .DESTINATION[DSC$B_SCALE] - .FLOAT_SCALE;
        WHILE .FLOAT_SCALE GTR 0 DO
            BEGIN
                TEMP_BUF1[LONG_1] = .TEMP_BUF1[S_LONG_1] / 2;
                FLOAT_SCALE = .FLOAT_SCALE - 1;
            END;
        IF .SIGN THEN TEMP_BUF1 = 0 - .TEMP_BUF1;
        OUTPUT [WORD_1] = .TEMP_BUF1 [S_WORD_1];
    END;
END;

[ DSC$K_DTYPE_B ]:
BEGIN
    ! If the target is not a binary scale, then just move the
    ! converted value in.
    IF NOT .DESTINATION [DSC$V_FL_BINSKALE]
    THEN
        CVTLB (TEMP_BUF2, .OUTPUT)
    ELSE
        ! If the sign and the scale of the H_Float are zero,
        ! then the value is zero.
        IF .INTMED_DATA[WORD_1] EQL 0
        THEN
            OUTPUT[BYTE_1] = 0
        ELSE
            BEGIN
                TEMP_BUF1 = 0;
                SIGN = .INTMED_DATA<15, 1, 0>;
                INTMED_DATA<15, 1, 0> = 0;
                FLOAT_SCALE = .INTMED_DATA[WORD_1] - 16384;
                IF .FLOAT_SCALE GTR (7 + .DESTINATION[DSC$B_SCALE])
                THEN
                    SIGNAL(DBG$_IINTOVF, 1, .DBG$GL_OPCODE_NAME)
                ELSE
                    BEGIN
                        TEMP_BUF1<6, 1, 0> = 1;
                        TEMP_BUF1<0, 6, 0> = .INTMED_DATA<26, 6, 0>;
                        FLOAT_SCALE = 7 + .DESTINATION[DSC$B_SCALE] - .FLOAT_SCALE;
                        WHILE .FLOAT_SCALE GTR 0 DO
                            BEGIN
                                TEMP_BUF1[LONG_1] = .TEMP_BUF1[S_LONG_1] / 2;
                                FLOAT_SCALE = .FLOAT_SCALE - 1;
                            END;
                        IF .SIGN THEN TEMP_BUF1 = 0 - .TEMP_BUF1;
                        OUTPUT [BYTE_1] = .TEMP_BUF1 [S_BYTE_1];
                    END;
            END;
        END;
    END;
END;

```

```

: 3815 3923 6
: 3816 3924 5
: 3817 3925 4
: 3818 3926 4
: 3819 3927 4
: 3820 3928 5
: 3821 3929 5
: 3822 3930 5
: 3823 3931 5
: 3824 3932 5
: 3825 3933 5
: 3826 3934 5
: 3827 3935 5
: 3828 3936 5
: 3829 3937 5
: 3830 3938 5
: 3831 3939 5
: 3832 3940 5
: 3833 3941 5
: 3834 3942 5
: 3835 3943 5
: 3836 3944 5
: 3837 3945 6
: 3838 3946 6
: 3839 3947 6
: 3840 3948 6
: 3841 3949 6
: 3842 3950 7
: 3843 3951 6
: 3844 3952 6
: 3845 3953 6
: 3846 3954 7
: 3847 3955 7
: 3848 3956 7
: 3849 3957 7
: 3850 3958 7
: 3851 3959 8
: 3852 3960 8
: 3853 3961 8
: 3854 3962 7
: 3855 3963 7
: 3856 3964 7
: 3857 3965 6
: 3858 3966 5
: 3859 3967 4
: 3860 3968 4
: 3861 3969 4
: 3862 3970 5
: 3863 3971 5
: 3864 3972 5
: 3865 3973 5
: 3866 3974 5
: 3867 3975 5
: 3868 3976 5
: 3869 3977 5
: 3870 3978 5
: 3871 3979 5

```

```

END;
END;
END;
[ DSC$K_DTYPE_W]:
BEGIN
    ! If the target is not a binary scale, then just move the
    ! converted value in.
    IF NOT .DESTINATION [DSC$V_FL_BINSCALE]
    THEN
        CVTLW (TEMP_BUF2, .OUTPUT)
    ELSE
        ! If the sign and the scale of the H_Float are zero,
        ! then the value is zero.
        IF .INTMED_DATA[WORD_1] EQL 0
        THEN
            OUTPUT[WORD_1] = 0
        ELSE
            BEGIN
                TEMP_BUF1 = 0;
                SIGN = .INTMED_DATA<15, 1, 0>;
                INTMED_DATA<15, 1, 0> = 0;
                FLOAT_SCALE = .INTMED_DATA[WORD_1] - 16384;
                IF .FLOAT_SCALE GTR (TS + .DESTINATION[DSC$B_SCALE])
                THEN
                    SIGNAL(DBG$_IINTOVF, 1, .DBG$GL_OPCODE_NAME)
                ELSE
                    BEGIN
                        TEMP_BUF1<14, 1, 0> = 1;
                        TEMP_BUF1<0, 14, 0> = .INTMED_DATA<18, 14, 0>;
                        FLOAT_SCALE = 15 + .DESTINATION[DSC$B_SCALE] - .FLOAT_SCALE;
                        WHILE .FLOAT_SCALE GTR 0 DO
                            BEGIN
                                TEMP_BUF1[LONG_1] = .TEMP_BUF1[S_LONG_1] / 2;
                                FLOAT_SCALE = .FLOAT_SCALE - 1;
                            END;
                        IF .SIGN THEN TEMP_BUF1 = 0 - .TEMP_BUF1;
                        OUTPUT [WORD_1] = .TEMP_BUF1 [S_WORD_1];
                    END;
                END;
            END;
        END;
    END;
[ DSC$K_DTYPE_L]:
BEGIN
    ! If the target is not a binary scale, then just move the
    ! converted value in.
    IF NOT .DESTINATION [DSC$V_FL_BINSCALE]
    THEN
        OUTPUT [LONG_1] = .TEMP_BUF2 [S_LONG_1]
    ELSE

```



```

3872 3980 5
3873 3981 5
3874 3982 5
3875 3983 5
3876 3984 5
3877 3985 5
3878 3986 5
3879 3987 6
3880 3988 6
3881 3989 6
3882 3990 6
3883 3991 6
3884 3992 7
3885 3993 6
3886 3994 6
3887 3995 6
3888 3996 7
3889 3997 7
3890 3998 7
3891 3999 7
3892 4000 7
3893 4001 7
3894 4002 8
3895 4003 8
3896 4004 8
3897 4005 7
3898 4006 7
3899 4007 7
3900 4008 6
3901 4009 5
3902 4010 4
3903 4011 4
3904 4012 4
3905 4013 5
3906 4014 5
3907 4015 5
3908 4016 5
3909 4017 5
3910 4018 5
3911 4019 6
3912 4020 6
3913 4021 5
3914 4022 4
3915 4023 4
3916 4024 4
3917 4025 4
3918 4026 4
3919 4027 3

```

```

! If the sign and the scale of the D_Float are zero,
! then the value is zero.
IF .TEMP_BUF1[WORD_1] EQL 0
THEN
    OUTPUT[LONG_1] = 0
ELSE
    BEGIN
        TEMP_BUF2 = 0;
        SIGN = .TEMP_BUF1<15, 1, 0>;
        TEMP_BUF1<15, 1, 0> = 0;
        FLOAT_SCALE = .TEMP_BUF1[WORD_1] - 16384;
        IF .FLOAT_SCALE GTR (31 + .DESTINATION[DSC$B_SCALE])
        THEN
            SIGNAL(DBG$IINTOVF, 1, .DBG$GL_OPCODE_NAME)
        ELSE
            BEGIN
                TEMP_BUF2<30, 1, 0> = 1;
                TEMP_BUF2<14, 16, 0> = .TEMP_BUF1<16, 16, 0>;
                TEMP_BUF2<0, 14, 0> = (.TEMP_BUF1+4)<18, 14, 0>;
                FLOAT_SCALE = 31 + .DESTINATION[DSC$B_SCALE] - .FLOAT_SCALE;
                WHILE .FLOAT_SCALE GTR 0 DO
                    BEGIN
                        TEMP_BUF2[LONG_1] = .TEMP_BUF2[S_LONG_1] / 2;
                        FLOAT_SCALE = .FLOAT_SCALE - 1;
                    END;
                IF .SIGN THEN TEMP_BUF2 = 0 - .TEMP_BUF2;
                OUTPUT [LONG_1] = .TEMP_BUF2 [S_LONG_1];
            END;
        END;
    END;

[DSC$K_DTYPE_V, DSC$K_DTYPE_SV, DSC$K_DTYPE_VU, DSC$K_DTYPE_SVU, DSC$K_DTYPE_TF]:
    BEGIN
        MAP
            OUTPUT: REF BITVECTOR[K OUTPUT BUFFER LENGTH * 8],
            INTMED_DATA: BITVECTOR[R_INTMED_DATA_LENGTH * 8];

        INCR I FROM 0 TO .DST_INFO[D_LEN] - 1 DO
            BEGIN
                OUTPUT[I] = .INTMED_DATA[I];
            END;
        END;

[INRANGE, OUTRANGE]:
    $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: nbds smlint');
    TES;
    !For NBDS_SMLINT
END;

```

```

3921 4028 3
3922 4029 3
3923 4030 3
3924 4031 3
3925 4032 3
3926 4033 3
3927 4034 4
3928 4035 4
3929 4036 4
3930 4037 4
3931 4038 4
3932 4039 4
3933 4040 4
3934 4041 4
3935 4042 4
3936 4043 4
3937 4044 4
3938 4045 4
3939 4046 4
3940 4047 4
3941 4048 4
3942 4049 4
3943 4050 3
3944 4051 3
3945 4052 3
3946 4053 4
3947 4054 4
3948 4055 4
3949 4056 4
3950 4057 4
3951 4058 4
3952 4059 4
3953 4060 3
3954 4061 3
3955 4062 3
3956 4063 4
3957 4064 4
3958 4065 4
3959 4066 4
3960 4067 4
3961 4068 4
3962 4069 4
3963 4070 4
3964 4071 4
3965 4072 4
3966 4073 4
3967 4074 4
3968 4075 4
3969 4076 4
3970 4077 4
3971 4078 4
3972 4079 4
3973 4080 4
3974 4081 4
3975 4082 4
3976 4083 4
3977 4084 4

```

```

[K_NBDS_LRGINT]:
CASE .DESTINATION [DSC$B_DTYPE] FROM DSC$K_DTYPE_LU TO DSC$K_DTYPE_O OF
SET
[DSC$K_DTYPE_LU]:
BEGIN
CLASS_S_DESC [DSC$W_LENGTH] = .SRC_INFO [S_LEN];
CLASS_S_DESC [DSC$A_POINTER] = .SRC_INFO [S_POINTER];
STATUS = OT$SCVT T D (CLASS_S_DESC, TEMP_BUF1, 0, -.SCALE,
(K_IGN_BLK$ OR K_ENB_UNDERFLOW OR K_IGN_TAB$ OR K_ENB_SCALE));
IF NOT .STATUS THEN SIGNAL (DBG$ INVNUMSTR, 1, .DBG$GL_OPCODE_NAME);
IF .TEMP_BUF1 < 15, 1, 0 THEN SIGNAL (DBG$ CVTNEGUNS, 1, .DBG$GL_OPCODE_NAME);
IF .CMPD (TEMP_BUF1, .LRGST D LU) GTR 0 THEN SIGNAL (DBG$ IINTOVF, 1, .DBG$GL_OPCODE_NAME);
BICPSW (%REF (%K_SET_ARITHMETIC_TRAP));
IF .CVT_ROUND_FLAG
THEN
CVTRDL (TEMP_BUF1, .OUTPUT)
ELSE
CVTDL (TEMP_BUF1, .OUTPUT);
BISPSW (%REF (%K_SET_ARITHMETIC_TRAP));
END;
[DSC$K_DTYPE_Q, DSC$K_DTYPE_QU]: ! M003
BEGIN
CLASS_S_DESC [DSC$W_LENGTH] = .SRC_INFO [S_LEN];
CLASS_S_DESC [DSC$A_POINTER] = .SRC_INFO [S_POINTER];
STATUS = OT$SCVT T H (CLASS_S_DESC, TEMP_BUF1, 0, -.SCALE,
(K_IGN_BLK$ OR K_ENB_UNDERFLOW OR K_IGN_TAB$ OR K_ENB_SCALE));
IF NOT .STATUS THEN SIGNAL (DBG$ INVNUMSTR, 1, .DBG$GL_OPCODE_NAME);
CVTRHQ (TEMP_BUF1, .OUTPUT)
END;
[DSC$K_DTYPE_O]: ! A004
BEGIN ! A004
LOCAL ! A004
Sign_flag : INITIAL( 0 ), ! A004
Current_character; ! A004
MAP OUTPUT : REF VECTOR[4]; ! A004
!++ ! A004
! Init the octaword ! A004
!-- ! A004
OUTPUT[ 3 ] = 0; ! A004
OUTPUT[ 2 ] = 0; ! A004
OUTPUT[ 1 ] = 0; ! A004
OUTPUT[ 0 ] = CH$RCHAR( .SRC_INFO[ S_POINTER ] ) - %C'0'; ! A004
!++ ! A004
! Test for bad characters ! A004
!-- ! A004
IF .OUTPUT[ 0 ] LSS 0 OR .OUTPUT[ 0 ] GTR 9 ! A004
THEN ! A004
IF .OUTPUT[ 0 ] EQL %C'-' - %C'0' ! A004
THEN ! A004

```

```

3978      4085 5      BEGIN
3979      4086 5      SRC_INFO[ S_POINTER ] = .SRC_INFO[ S_POINTER ] + 1;
3980      4087 5      SRC_INFO[ S_LEN ] = .SRC_INFO[ S_LEN ] - 1;
3981      4088 5      OUTPUT[ 0 ] = CH$RCHAR( .SRC_INFO[ S_POINTER ] ) -
3982      4089 5      %C'0';
3983      4090 5      Sign_flag = 1;
3984      4091 5      END
3985      4092 4      ELSE
3986      4093 4      SIGNAL( DBG$_INVDIGDEC, 2, 1, .SRC_INFO[ S_POINTER ] );
3987      4094 4
3988      4095 4      ++
3989      4096 4      For each character
3990      4097 4      multiply by 10
3991      4098 4      add it to the low order long word
3992      4099 4      --
3993      4100 4      INCR Current_char_num FROM 1 TO .SRC_INFO[ S_LEN ] - 1 DO
3994      4101 5      BEGIN
3995      4102 5      DBG$CVT_SCALE_OU_UP_BY_10 R1( .OUTPUT );
3996      4103 5      Current_character = CH$RCHAR( .SRC_INFO[ S_POINTER ] +
3997      4104 5      .Current_char_num ) -
3998      4105 5      %C'0';
3999      4106 5
4000      4107 5      ++
4001      4108 5      Test for bad characters
4002      4109 5      --
4003      4110 5      IF ( .Current_character LSS 0 ) OR
4004      4111 6      ( .Current_character GTR 9 )
4005      4112 5      THEN
4006      4113 5      SIGNAL( DBG$_INVDIGDEC, 2, 1, .SRC_INFO[ S_POINTER ] +
4007      4114 5      .Current_char_num );
4008      4115 5
4009      4116 5      OUTPUT[ 0 ] = .OUTPUT[ 0 ] + .Current_character;
4010      4117 4      END;
4011      4118 4
4012      4119 4      ++
4013      4120 4      When there was a negative we subtract from 0
4014      4121 4      --
4015      4122 4      IF .Sign_flag
4016      4123 4      THEN
4017      4124 5      BEGIN
4018      4125 5      LOCAL
4019      4126 5      Octaword_zero : VECTOR[4];
4020      4127 5
4021      4128 5      Octaword_zero[ 3 ] = 0;
4022      4129 5      Octaword_zero[ 2 ] = 0;
4023      4130 5      Octaword_zero[ 1 ] = 0;
4024      4131 5      Octaword_zero[ 0 ] = 0;
4025      4132 5
4026      4133 5      SUBM( 4, .OUTPUT, Octaword_zero, .OUTPUT );
4027      4134 5
4028      4135 4      END;
4029      4136 4
4030      4137 3      END;
4031      4138 3
4032      4139 3      [INRANGE, OUTRANGE]:
4033      4140 3      $DBG_ERROR ( 'DBGCVTDX\DBG$CVT_DX_DX: nbds_lrgint' );
4034      4141 3      TES;
                                ! For NBDS_LRGINT
```

```

: 4036      4142  3
: 4037      4143  3
: 4038      4144  4
: 4039      4145  4
: 4040      4146  4
: 4041      4147  4
: 4042      4148  4
: 4043      4149  4
: 4044      4150  4
: 4045      4151  5
: 4046      4152  5
: 4047      4153  5
: 4048      4154  5
: 4049      4155  5
: 4050      4156  5
: 4051      4157  5
: 4052      4158  5
: 4053      4159  6
: 4054      4160  6
: 4055      4161  6
: 4056      4162  6
: 4057      4163  6
: 4058      4164  6
: 4059      4165  6
: 4060      4166  5
: 4061      4167  4
: 4062      4168  4
: 4063      4169  4
: 4064      4170  5
: 4065      4171  5
: 4066      4172  5
: 4067      4173  5
: 4068      4174  5
: 4069      4175  5
: 4070      4176  5
: 4071      4177  5
: 4072      4178  5
: 4073      4179  5
: 4074      4180  5
: 4075      4181  5
: 4076      4182  4
: 4077      4183  4
: 4078      4184  4
: 4079      4185  4
: 4080      4186  4
: 4081      4187  3

```

```

[K_NBDS_LRGFLT_CMLX]:
  BEGIN
    CLASS_S_DESC [DSC$W_LENGTH] = .SRC_INFO [S_LEN];
    CLASS_S_DESC [DSC$A_POINTER] = .SRC_INFO [S_POINTER];
    CASE .DESTINATION [DSC$B_DTYPE] FROM DSC$K_DTYPE_G TO DSC$K_DTYPE_HC OF
      SET
        [DSC$K_DTYPE_G, DSC$K_DTYPE_GC]:
          BEGIN
            STATUS = OTSS$CVT T_G (CLASS S_DESC, TEMP_BUF1, 0, -.SCALE,
              (K_IGN_BLK$ OR K_ENB_UNDERFLOW OR K_IGN_TAB$ OR K_ENB_SCALE));
            IF NOT .STATUS THEN SIGNAL (DBG$ INVNUMSTR, -1, .DBG$GC_OPCODE_NAME);
            OUTPUT [LONG_1] = .TEMP_BUF1 [LONG_1];
            OUTPUT [LONG_2] = .TEMP_BUF1 [LONG_2];
            IF .DESTINATION [DSC$B_DTYPE] EQL DSC$K_DTYPE_GC
              THEN
                BEGIN
                  ! Fill in imaginary part with 0;
                  !
                  OUTPUT [LONG_3] = 0;
                  OUTPUT [LONG_4] = 0;
                END;
            END;
          [DSC$K_DTYPE_H, DSC$K_DTYPE_HC]:
            BEGIN
              STATUS = OTSS$CVT T_H (CLASS S_DESC, TEMP_BUF1, 0, -.SCALE,
                (K_IGN_BLK$ OR K_ENB_UNDERFLOW OR K_IGN_TAB$ OR K_ENB_SCALE));
              IF NOT .STATUS THEN SIGNAL (DBG$ INVNUMSTR, -1, .DBG$GC_OPCODE_NAME);
              CH$MOVE (16, TEMP_BUF1, .OUTPUT);
              IF .DESTINATION [DSC$B_DTYPE] EQL DSC$K_DTYPE_HC
                THEN
                  ! Fill in imaginary part with 0.
                  !
                  CH$FILL (0, 16, .OUTPUT+16);
            END;
          [INRANGE, OUTRANGE]:
            $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: nbds_lrgflt_cmlx');
          TES;
            ! For NBDS_LRGFLT_CMLX
        END;
    END;

```

```

: 4083 4188 3
: 4084 4189 3
: 4085 4190 3
: 4086 4191 3
: 4087 4192 3
: 4088 4193 3
: 4089 4194 4
: 4090 4195 4
: 4091 4196 4
: 4092 4197 5
: 4093 4198 5
: 4094 4199 5
: 4095 4200 5
: 4096 4201 5
: 4097 4202 5
: 4098 4203 5
: 4099 4204 6
: 4100 4205 6
: 4101 4206 6
: 4102 4207 6
: 4103 4208 6
: 4104 4209 6
: 4105 4210 6
: 4106 4211 6
: 4107 4212 6
: 4108 4213 6
: 4109 4214 6
: 4110 4215 6
: 4111 4216 6
: 4112 4217 6
: 4113 4218 6
: 4114 4219 6
: 4115 4220 6
: 4116 4221 7
: 4117 4222 7
: 4118 4223 7
: 4119 4224 7
: 4120 4225 7
: 4121 4226 7
: 4122 4227 7
: 4123 4228 7
: 4124 4229 7
: 4125 4230 6
: 4126 4231 6
: 4127 4232 6
: 4128 4233 7
: 4129 4234 7
: 4130 4235 7
: 4131 4236 7
: 4132 4237 7
: 4133 4238 7
: 4134 4239 7
: 4135 4240 7
: 4136 4241 7
: 4137 4242 6
: 4138 4243 6
: 4139 4244 6

```

```

[K_NBDS_NBDS]:
  SELECTONE .DESTINATION [DSC$B_DTYPE] OF
  SET
    [DSC$K_DTYPE_BU, DSC$K_DTYPE_T, DSC$K_DTYPE_VT, .SC$K_DTYPE_AC, DSC$K_DTYPE_AZ]:
    BEGIN
      IF .SCALE NEQ 0
      THEN
        BEGIN
          CLASS_S_DESC [DSC$W_LENGTH] = .SRC_INFO [S_LEN];
          CLASS_S_DESC [DSC$A_POINTER] = .SRC_INFO [S_POINTER];
          STATUS = OT$SCVT T_R (CLASS_S_DESC, TEMP_BUF1, 0, -.SCALE,
            (K_IGN_BLKS OR K_ENB_UNDERFLOW OR K_IGN_TABS OR K_ENB_SCALE));
          IF .STATUS
          THEN
            BEGIN
              CLASS_S_DESC [DSC$W_LENGTH] = K_TEMP_BUF_LENGTH;
              CLASS_S_DESC [DSC$A_POINTER] = TEMP_BUF2;
              IF .DST_INFO [D_LEN] - 9 LEQ 0
              THEN
                DIGITS_IN_FRACT = 33
              ELSE
                DIGITS_IN_FRACT = MIN (33, .DST_INFO [D_LEN] - 9);
              STATUS = FOR$CVT H TE (TEMP_BUF1, CLASS_S_DESC, .DIGITS_IN_FRACT, 0, 0, 4);
              IF NOT .STATUS THEN $DBG_ERROR ('DBGCVTDX\DBG$CVT DX_DX: error in h-to-te conve
                BUF_OFFSET = CH$FIND NOT_CH (K_TEMP_BUF_LENGTH, TEMP_BUF2, %C' ') - TEMP_BUF2;
                FINAL_LEN = K_TEMP_BUF_LENGTH - .BUF_OFFSET;
                OUTPUT_STR_LEN = .FINAL_LEN;
            END
          SELECTONE .DESTINATION[DSC$B_DTYPE] OF
          SET
            [DSC$K_DTYPE_AC]:
            BEGIN
              MAP
                OUTPUT: REF VECTOR[, BYTE];
                CLASS_S_DESC[DSC$W_LENGTH] = .FINAL_LEN;
                CLASS_S_DESC[DSC$A_POINTER] = OUTPUT[1];
                STATUS = LIB$SCOPY_R DX6 (.FINAL_LEN, TEMP_BUF2 + .BUF_OFFSET, CLASS_S_D
                IF .STATUS EQL LIB$STRTRU THEN SIGNAL (DBG$_ISTRTRU, T, .DBG$GL_OPCODE_
                IF NOT .STATUS THEN SIGNAL (.STATUS);
                OUTPUT[0] = .FINAL_LEN;
                END;
            [DSC$K_DTYPE_AZ]:
            BEGIN
              MAP
                OUTPUT: REF VECTOR[, BYTE];
                CLASS_S_DESC[DSC$W_LENGTH] = .FINAL_LEN;
                CLASS_S_DESC[DSC$A_POINTER] = OUTPUT[0];
                STATUS = LIB$SCOPY_R DX6 (.FINAL_LEN, TEMP_BUF2 + .BUF_OFFSET, CLASS_S_D
                IF .STATUS EQL LIB$STRTRU THEN SIGNAL (DBG$_ISTRTRU, T, .DBG$GL_OPCODE_
                IF NOT .STATUS THEN SIGNAL (.STATUS);
                OUTPUT[.FINAL_LEN + 1] = 0;
                END;
          [OTHERWISE]:

```

```

: 4140
: 4141
: 4142
: 4143
: 4144
: 4145
: 4146
: 4147
: 4148
: 4149
: 4150
: 4151
: 4152
: 4153
: 4154
: 4155
: 4156
: 4157
: 4158
: 4159
: 4160
: 4161
: 4162
: 4163
: 4164
: 4165
: 4166
: 4167
: 4168
: 4169
: 4170
: 4171
: 4172
: 4173
: 4174
: 4175
: 4176
: 4177
: 4178
: 4179
: 4180
: 4181
: 4182
: 4183
: 4184
: 4185
: 4186
: 4187
: 4188
: 4189
: 4190
: 4191
: 4192
: 4193
: 4194
: 4195
: 4196

```

```

4245 7
4246 7
4247 7
4248 7
4249 6
4250 6
4251 6
4252 5
4253 5
4254 5
4255 4
4256 5
4257 5
4258 5
4259 5
4260 5
4261 6
4262 6
4263 6
4264 6
4265 6
4266 6
4267 6
4268 6
4269 6
4270 5
4271 5
4272 5
4273 6
4274 6
4275 6
4276 6
4277 6
4278 6
4279 6
4280 6
4281 6
4282 5
4283 5
4284 5
4285 6
4286 6
4287 6
4288 6
4289 5
4290 5
4291 4
4292 3
4293 3
4294 3
4295 4
4296 4
4297 4
4298 4
4299 4
4300 4
4301 4

```

```

BEGIN
STATUS = LIB$SCOPY R DX6 (.FINAL LEN, TEMP BUF2 + .BUF OFFSET, .DESTINAT
IF .STATUS EQL LIB$ STRTRU THEN SIGNAL (DBG$_ISTRTRU, T, .DBG$GL_OPCODE_
IF NOT .STATUS THEN SIGNAL (.STATUS);
END;
TES;
END
ELSE
SIGNAL (DBG$_INVNUMSTR, 1, .DBG$GL_OPCODE_NAME);
END
ELSE
BEGIN
OUTPUT STR_LEN = .SOURCE [DSC$W_LENGTH];
SELECTIONE .DESTINATION[DSC$B_DTYPE] OF
SET
[DSC$K_DTYPE_AC]:
BEGIN
MAP
OUTPUT: REF VECTOR[, BYTE];
CLASS_S_DESC[DSC$W_LENGTH] = .SOURCE[DSC$W_LENGTH];
CLASS_S_DESC[DSC$A_POINTER] = OUTPUT[1];
STATUS = LIB$SCOPY DXDX6 (.SOURCE, CLASS S DESC);
IF .STATUS EQL LIB$ STRTRU THEN SIGNAL (DBG$_ISTRTRU, 1, .DBG$GL_OPCODE_NAME
IF NOT .STATUS THEN SIGNAL (.STATUS);
OUTPUT[0] = .SOURCE[DSC$W_LENGTH];
END;
[DSC$K_DTYPE_AZ]:
BEGIN
MAP
OUTPUT: REF VECTOR[, BYTE];
CLASS_S_DESC[DSC$W_LENGTH] = .SOURCE[DSC$W_LENGTH];
CLASS_S_DESC[DSC$A_POINTER] = OUTPUT[0];
STATUS = LIB$SCOPY DXDX6 (.SOURCE, CLASS S DESC);
IF .STATUS EQL LIB$ STRTRU THEN SIGNAL (DBG$_ISTRTRU, 1, .DBG$GL_OPCODE_NAME
IF NOT .STATUS THEN SIGNAL (.STATUS);
OUTPUT[.SOURCE[DSC$W_LENGTH]] = 0;
END;
[OTHERWISE]:
BEGIN
STATUS = LIB$SCOPY DXDX6 (.SOURCE, .DESTINATION);
IF .STATUS EQL LIB$ STRTRU THEN SIGNAL (DBG$_ISTRTRU, 1, .DBG$GL_OPCODE_NAME
IF NOT .STATUS THEN SIGNAL (.STATUS);
END;
TES;
END;
END;
[DSC$K_DTYPE_ZI]:
BEGIN
OWN
INPUT STR: VECTOR[100, BYTE];
OUTPUT STR: VECTOR[100, BYTE];
INPUT STR[0] = .SRC INFO[S_LEN];
CH$MOVE (.INPUT STR[0], .SRC INFO[S_POINTER], INPUT STR[1]);
STATUS = DBG$INS_ENCODE (INPUT STR, OUTPUT STR, .DESTINATION[DSC$A_POINTER]);

```

```

4197      4302      4      IF NOT .STATUS THEN SIGNAL (.STATUS);
4198      4303      4      DESTINATION[DSC$W_LENGTH] = .OUTPUT_STR[0];
4199      4304      4      CH$MOVE (.OUTPUT_STR[0], OUTPUT_STR[1], .DESTINATION[DSC$A_POINTER]);
4200      4305      3      END;
4201      4306      3
4202      4307      3      [OTHERWISE]:
4203      4308      3      $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: nbds_nbd');
4204      4309      3      TES;
4205      4310      3      !For NBDS_NBDS
4206      4311      3      TES;
4207      4312      3      !End of the main CASE statement.
4208      4313      3
4209      4314      3      ! If the destination class is unaligned, then the output will be in a temporary buffer.
4210      4315      3      ! Copy from the temporary buffer to: (.destination pointer + number of bits to be offset).
4211      4316      3
4212      4317      3      IF .DESTINATION[DSC$B_CLASS] EQL DSC$K_CLASS_UBS
4213      4318      3      THEN
4214      4319      4      BEGIN
4215      4320      4      SRC_POS = 0;
4216      4321      4      DST_POS = .DESTINATION[DSC$L_POS];
4217      4322      4      CASE .DESTINATION[DSC$B_DTYPE] FROM K_MIN_DTYPE_STA TO K_MAX_DTYPE_STA OF
4218      4323      4      SET
4219      4324      4      [DSC$K_DTYPE_V, DSC$K_DTYPE_SV, DSC$K_DTYPE_VU, DSC$K_DTYPE_SVU, DSC$K_DTYPE_TF]:
4220      4325      5      BEGIN
4221      4326      5      MAP
4222      4327      5      DESTINATION_PTR: REF BITVECTOR,
4223      4328      5      OUTPUT: REF BITVECTOR;
4224      4329      5
4225      4330      5      INCR I FROM 1 TO .DESTINATION[DSC$W_LENGTH] DO
4226      4331      6      BEGIN
4227      4332      6      DESTINATION_PTR[DST_POS] = .OUTPUT[SRC_POS];
4228      4333      6      DST_POS = .DST_POS + 1;
4229      4334      6      SRC_POS = .SRC_POS + 1;
4230      4335      5      END;
4231      4336      4      END;
4232      4337      4
4233      4338      4      [INRANGE]:
4234      4339      5      BEGIN
4235      4340      5      INCR I FROM 1 TO .DESTINATION[DSC$W_LENGTH] DO
4236      4341      6      BEGIN
4237      4342      6      (.DESTINATION_PTR)<.DST_POS, 8> = (.OUTPUT)<.SRC_POS, 8>;
4238      4343      6      DST_POS = .DST_POS + 8;
4239      4344      6      SRC_POS = .SRC_POS + 8;
4240      4345      5      END;
4241      4346      4      END;
4242      4347      4
4243      4348      4      [OUTRANGE]:
4244      4349      4      $DBG_ERROR ('DBGCVTDX\DBG$CVT_DX_DX: invalid dtype');
4245      4350      4      TES;
4246      4351      3      END;
4247      4352      3
4248      4353      2      END;
4249      4354      2      ! End the ELSE part of the Absolute Date Time IF
4250      4355      2      ! If output string length is requested then supply it.
4251      4356      2
4252      4357      2      IF ACTUALCOUNT() GTR 2 THEN (.OUTLEN)<0, 16, 0> = .OUTPUT_STR_LEN;
4253      4358      2

```

: 4254 4359 1 END;

! End of routine DBG\$CVT\_DX\_DX.

```
.PSECT DBG$PLIT,NOWRT, SHR, PIC,0

                                00 00 5C 29 67 49 29 04 00AD2 .BLKB 2
                                0000FF00 FFFF507F 00AD4 P.ADX: .ASCII <4>\)Ig)\<92><0><0>
                                0000FFFE FFFF4020 00ADC P.ADY: .LONG ^XFFFF507F, ^X0000FF00
                                00000000 00000000 00AE4 P.ADZ: .LONG ^XFFFF4020, ^X0000FFFE, ^X00000000, ^X0000-
                                                                0000
43 24 47 42 44 5C 58 44 54 56 43 00 00 00 0C 00AF4 P.AEA: .ASCII <12><0><0><0>
61 76 6E 69 20 20 3A 58 44 5F 58 47 42 44 34 00AF8 P.AEB: .ASCII \4DBGCVTDX\<92>\DBG$CVT_DX_DX: invalid \
                                                                20 64 69 6C 00B07
69 72 63 73 65 64 20 6E 69 20 65 70 79 74 64 00B16 .ASCII \dtype in descriptor\
                                                                72 6F 74 70 00B1A
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 34 00B29 P.AEC: .ASCII \4DBGCVTDX\<92>\DBG$CVT_DX_DX: invalid \
61 76 6E 69 20 20 3A 58 44 5F 58 44 5F 54 56 00B2D
                                                                20 64 69 6C 00B3C
69 72 63 73 65 64 20 6E 69 20 73 73 61 6C 63 00B48 .ASCII \class in descriptor\
                                                                72 6F 74 70 00B5E
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 38 00B62 P.AED: .ASCII \8DBGCVTDX\<92>\DBG$CVT_DX_DX: invalid \
61 76 6E 69 20 20 3A 58 44 5F 58 44 5F 54 56 00B71
                                                                20 64 69 6C 00B80
6D 6F 63 20 65 70 79 74 64 2D 73 73 61 6C 63 00B84 .ASCII \class-dtype combination\
                                                                6E 6F 69 74 61 6E 69 62 00B93
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 39 00B98 P.AEE: .ASCII \9DBGCVTDX\<92>\DBG$CVT_DX_DX: invalid \
61 76 6E 69 20 20 3A 58 44 5F 58 44 5F 54 56 00BAA
                                                                20 64 69 6C 00BB9
74 73 20 65 74 79 62 20 63 69 72 65 6D 75 6E 00BBD .ASCII \numeric byte string data\
                                                                61 74 61 64 20 67 6E 69 72 00BCC
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 26 00BD5 P.AEF: .ASCII \8DBGCVTDX\<92>\DBG$CVT_DX_DX: smlint_s\
69 6C 6D 73 20 20 3A 58 44 5F 58 44 5F 54 56 00BE4
                                                                73 5F 74 6E 00BF3
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 26 00BF7 P.AEG: .ASCII \mlint\
69 6C 6D 73 20 20 3A 58 44 5F 58 44 5F 54 56 00BFC .ASCII \8DBGCVTDX\<92>\DBG$CVT_DX_DX: smlint_l\
                                                                6C 5F 74 6E 00C0B
                                                                74 6E 69 67 72 00C1A
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 26 00C1E P.AEH: .ASCII \rgint\
69 67 72 6C 20 20 3A 58 44 5F 58 44 5F 54 56 00C23 .ASCII \8DBGCVTDX\<92>\DBG$CVT_DX_DX: lrgint_l\
                                                                6C 5F 74 6E 00C32
                                                                74 6E 69 67 72 00C41
                                                                00000000 00004100 00C4A .ASCII \rgint\
                                                                00000000 00004100 00C4A .BLKB 2
                                                                00000000 00004100 00C4C P.AEI: .LONG ^X00004100, ^X00000000
                                                                00000000 00004100 00C54 P.AEJ: .LONG ^X00004100, ^X00000000
                                                                00000000 00004220 00C5C P.AEK: .LONG ^X00004220, ^X00000000
                                                                00000000 00004220 00C64 P.AEL: .LONG ^X00004220, ^X00000000
                                                                00000000 00004100 00C6C P.AEM: .LONG ^X00004100, ^X00000000
                                                                00000000 00004100 00C74 P.AEN: .LONG ^X00004100, ^X00000000
                                                                00000000 00004220 00C7C P.AEO: .LONG ^X00004220, ^X00000000
                                                                00000000 00004220 00C84 P.AEP: .LONG ^X00004220, ^X00000000
                                                                00000000 00004100 00C8C P.AEQ: .LONG ^X00004100, ^X00000000
                                                                00000000 00004100 00C94 P.AER: .LONG ^X00004100, ^X00000000
                                                                00000000 00004100 00C9C P.AES: .LONG ^X00004100, ^X00000000
```



										00000000	00004100	00CA4	P.AET:	.LONG	^X00004100, ^X00000000			
										00000000	00004220	00CAC	P.AEU:	.LONG	^X00004220, ^X00000000			
										00000000	00004220	00CB4	P.AEV:	.LONG	^X00004220, ^X00000000			
										00000000	00004220	00CBC	P.AEW:	.LONG	^X00004220, ^X00000000			
										00000000	00004220	00CC4	P.AEX:	.LONG	^X00004220, ^X00000000			
43	24	47	42	44	5C	58	44	54	56	43	47	42	44	2B	00CCC	P.AEY:	.ASCII	\+DBGCVTDX\<92>\DBG\$CVT_DX_DX: smlint_s\
69	6C	6D	73	20	20	3A	58	44	5F	58	44	5F	54	56	00CDB			
											73	5F	74	6E	00CEA			
															00CEE			
43	24	47	42	44	5C	58	44	54	56	43	47	42	44	2B	00CF8	P.AEZ:	.ASCII	\mlfltcmplx\
69	67	72	6C	20	20	3A	58	44	5F	58	44	5F	54	56	00D07		.ASCII	\+DBGCVTDX\<92>\DBG\$CVT_DX_DX: lrgint_s\
											73	5F	74	6E	00D16			
															00D1A			
43	24	47	42	44	5C	58	44	54	56	43	47	42	44	30	00D24	P.AFA:	.ASCII	\mlfltcmplx\
66	6C	6D	73	20	20	3A	58	44	5F	58	44	5F	54	56	00D33		.ASCII	\0DBGCVTDX\<92>\DBG\$CVT_DX_DX: smlfltcm\
											6D	63	74	6C	00D42			
78	6C	70	6D	63	74	6C	66	6C	6D	73	5F	78	6C	70	00D46			
43	24	47	42	44	5C	58	44	54	56	43	47	42	44	28	00D55	P.AFB:	.ASCII	\plx_smlfltcmplx\
5F	63	65	64	20	20	3A	58	44	5F	58	44	5F	54	56	00D64		.ASCII	\(DBGCVTDX\<92>\DBG\$CVT_DX_DX: dec_smlf\
											66	6C	6D	73	00D73			
											70	6D	63	74	6C	00D77		
43	24	47	42	44	5C	58	44	54	56	43	47	42	44	29	00D7E	P.AFC:	.ASCII	\ltcmplx\
73	64	62	6E	20	20	3A	58	44	5F	58	44	5F	54	56	00D8D		.ASCII	\)DBGCVTDX\<92>\DBG\$CVT_DX_DX: nbds_sml\
											6C	6D	73	5F	00D9C			
											6D	63	74	6C	66	00DA0		
					00000000		00000000		00000000		00000000		00004002		00DA8	P.AFD:	.ASCII	\fltcmplx\
																.LONG	-	
																	^X00004002, ^X00000000, ^X00000000, ^X0000-	
																	0000	
					00000000		00000000		00000000		00004002		00DB8	P.AFE:	.LONG	-		
																	^X00004002, ^X00000000, ^X00000000, ^X0000-	
																	0000	
					00000000		00000000		00000000		40004004		00DC8	P.AFF:	.LONG	-		
																	^X40004004, ^X00000000, ^X00000000, ^X0000-	
																	0000	
					00000000		00000000		00000000		40004004		00DD8	P.AFG:	.LONG	-		
																	^X40004004, ^X00000000, ^X00000000, ^X0000-	
																	0000	
					00000000		00000000		00000000		00004002		00DE8	P.AFH:	.LONG	-		
																	^X00004002, ^X00000000, ^X00000000, ^X0000-	
																	0000	
					00000000		00000000		00000000		00004002		00DF8	P.AFI:	.LONG	-		
																	^X00004002, ^X00000000, ^X00000000, ^X0000-	
																	0000	
					00000000		00000000		00000000		40004004		00E08	P.AFJ:	.LONG	-		
																	^X40004004, ^X00000000, ^X00000000, ^X0000-	
																	0000	
					00000000		00000000		00000000		40004004		00E18	P.AFK:	.LONG	-		
																	^X40004004, ^X00000000, ^X00000000, ^X0000-	
																	0000	
					00000000		00000000		00000000		00004002		00E28	P.AFL:	.LONG	-		
																	^X00004002, ^X00000000, ^X00000000, ^X0000-	
																	0000	
					00000000		00000000		00000000		00004002		00E38	P.AFM:	.LONG	-		
																	^X00004002, ^X00000000, ^X00000000, ^X0000-	
																	0000	
					00000000		00000000		00000000		00004002		00E48	P.AFN:	.LONG	-		
																	^X00004002, ^X00000000, ^X00000000, ^X0000-	

00000000	00000000	00000000	00004002	00E58 P.AFD:	.LONG	0000 - ^X00004002, ^X00000000, ^X00000000, ^X0000- 0000
00000000	00000000	00000000	40004004	00E68 P.AFP:	.LONG	- ^X40004004, ^X00000000, ^X00000000, ^X0000- 0000
00000000	00000000	00000000	40004004	00E78 P.AFQ:	.LONG	- ^X40004004, ^X00000000, ^X00000000, ^X0000- 0000
00000000	00000000	00000000	40004004	00E88 P.AFR:	.LONG	- ^X40004004, ^X00000000, ^X00000000, ^X0000- 0000
00000000	00000000	00000000	40004004	00E98 P.AFS:	.LONG	- ^X40004004, ^X00000000, ^X00000000, ^X0000- 0000
00000000	00000000	00000000	00004002	00EA8 P.AFT:	.LONG	- ^X00004002, ^X00000000, ^X00000000, ^X0000- 0000
00000000	00000000	00000000	00004002	00EB8 P.AFU:	.LONG	- ^X00004002, ^X00000000, ^X00000000, ^X0000- 0000
00000000	00000000	00000000	00004002	00EC8 P.AFV:	.LONG	- ^X00004002, ^X00000000, ^X00000000, ^X0000- 0000
00000000	00000000	00000000	00004002	00ED8 P.AFW:	.LONG	- ^X00004002, ^X00000000, ^X00000000, ^X0000- 0000
00000000	00000000	00000000	40004004	00EE8 P.AFX:	.LONG	- ^X40004004, ^X00000000, ^X00000000, ^X0000- 0000
00000000	00000000	00000000	40004004	00EF8 P.AFY:	.LONG	- ^X40004004, ^X00000000, ^X00000000, ^X0000- 0000
00000000	00000000	00000000	40004004	00F08 P.AFZ:	.LONG	- ^X40004004, ^X00000000, ^X00000000, ^X0000- 0000
00000000	00000000	00000000	40004004	00F18 P.AGA:	.LONG	- ^X40004004, ^X00000000, ^X00000000, ^X0000- 0000
00000000	00000000	00000000	00004002	00F28 P.AGB:	.LONG	- ^X00004002, ^X00000000, ^X00000000, ^X0000- 0000
00000000	00000000	00000000	00004002	00F38 P.AGC:	.LONG	- ^X00004002, ^X00000000, ^X00000000, ^X0000- 0000
00000000	00000000	00000000	00004002	00F48 P.AGD:	.LONG	- ^X00004002, ^X00000000, ^X00000000, ^X0000- 0000
00000000	00000000	00000000	00004002	00F58 P.AGE:	.LONG	- ^X00004002, ^X00000000, ^X00000000, ^X0000- 0000
00000000	00000000	00000000	40004004	00F68 P.AGF:	.LONG	- ^X40004004, ^X00000000, ^X00000000, ^X0000- 0000
00000000	00000000	00000000	40004004	00F78 P.AGG:	.LONG	- ^X40004004, ^X00000000, ^X00000000, ^X0000-

```
00000000 00000000 00000000 40004004 00F88 P.AGH: .LONG 0000
^x40004004, ^x00000000, ^x00000000, ^x0000-
0000
00000000 00000000 00000000 40004004 00F98 P.AGI: .LONG
^x40004004, ^x00000000, ^x00000000, ^x0000-
0000
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 2B 00FA8 P.AGJ: .ASCII \+DBGCVTDX\<92>\DBG$CVT_DX_DX: smlint_l\
69 6C 6D 73 20 20 3A 58 44 5F 58 44 5F 54 56 00FB7
6C 5F 74 6E 00FC6
78 6C 70 6D 63 74 6C 66 67 72 00FCA
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 2B 00FD4 P.AGK: .ASCII \rgfltcmlx\
69 67 72 6C 20 20 3A 58 44 5F 58 44 5F 54 56 00FE3
6C 5F 74 6E 00FF2
78 6C 70 6D 63 74 6C 66 67 72 00FF6 P.AGL: .ASCII \rgfltcmlx\
66 6C 6D 73 20 20 3A 58 44 5F 58 44 5F 54 56 01000 P.AGL: .ASCII \DBGCVTDX\<92>\DBG$CVT_DX_DX: smlfltcml\
6C 6D 63 74 6C 0100F
78 6C 70 6D 63 74 6C 70 0101E
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 30 01022 P.AGM: .ASCII \plx_lrgfltcmlx\
66 67 72 6C 20 20 3A 58 44 5F 58 44 5F 54 56 01031 P.AGM: .ASCII \DBGCVTDX\<92>\DBG$CVT_DX_DX: lrgfltcml\
6C 6D 63 74 6C 01040
78 6C 70 6D 63 74 6C 70 0104F
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 28 01053 P.AGN: .ASCII \plx_lrgfltcmlx\
5F 63 65 64 20 20 3A 58 44 5F 58 44 5F 54 56 01062 P.AGN: .ASCII \DBGCVTDX\<92>\DBG$CVT_DX_DX: dec_lrgf\
6C 67 72 6C 01071
78 6C 70 6D 63 74 6C 01080
6C 67 72 6C 01084
00 00 00 2C 01088 .ASCII \ltcmlx\
00 00 00 2C 0108C P.AGO: .ASCII 1
00 00 00 2C 01090 P.AGP: .ASCII \, \<0><0><0>
00 00 00 2C 01094 P.AGQ: .ASCII \, \<0><0><0>
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 23 010A3 P.AGQ: .ASCII \DBGCVTDX\<92>\DBG$CVT_DX_DX: smlint_d\
69 6C 6D 73 20 20 3A 58 44 5F 58 44 5F 54 56 010B2
6C 6D 63 74 6C 010B6
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 20 010B8 P.AGR: .ASCII \ec\
5F 63 65 64 20 20 3A 58 44 5F 58 44 5F 54 56 010C7 P.AGR: .ASCII \DBGCVTDX\<92>\DBG$CVT_DX_DX: dec_dec\
6C 6D 63 74 6C 010D6
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 26 010D9 P.AGS: .ASCII \&DBGCVTDX\<92>\DBG$CVT_DX_DX: lrgint_s\
69 67 72 6C 20 20 3A 58 44 5F 58 44 5F 54 56 010E8
73 5F 74 6E 010F7
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 23 010FB P.AGT: .ASCII \mlint\
69 67 72 6C 20 20 3A 58 44 5F 58 44 5F 54 56 01100 P.AGT: .ASCII \DBGCVTDX\<92>\DBG$CVT_DX_DX: lrgint_d\
6C 6D 63 74 6C 0110F
6C 6D 63 74 6C 0111E
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 28 01122 P.AGU: .ASCII \ec\
66 6C 6D 73 20 20 3A 58 44 5F 58 44 5F 54 56 01124 P.AGU: .ASCII \DBGCVTDX\<92>\DBG$CVT_DX_DX: smlfltcml\
6C 6D 63 74 6C 01133
6C 6D 63 74 6C 01142
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 28 01146 P.AGV: .ASCII \plx_dec\
66 67 72 6C 20 20 3A 58 44 5F 58 44 5F 54 56 0115D P.AGV: .ASCII \DBGCVTDX\<92>\DBG$CVT_DX_DX: lrgfltcml\
6C 6D 63 74 6C 0115C
6C 6D 63 74 6C 0116B
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 21 0116F P.AGW: .ASCII \plx_dec\
73 64 62 6E 20 20 3A 58 44 5F 58 44 5F 54 56 01176 P.AGW: .ASCII \!DBGCVTDX\<92>\DBG$CVT_DX_DX: nbds_dec\
6C 6D 63 74 6C 01185
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 31 01194 P.AGX: .ASCII \DBGCVTDX\<92>\DBG$CVT_DX_DX inconsistent\
69 6C 6D 73 20 20 3A 58 44 5F 58 44 5F 54 56 01198
```

```
73 6E 6F 63 6E 69 20 58 44 5F 58 44 5F 54 56 011A7
72 6F 74 63 61 66 20 65 6C 61 63 65 74 73 69 011B6
                                011BA
                                011C9
                                011CA
                                011CC P.AGY: .BLKB 2
                                011D4 P.AGZ: .LONG ^x00004100, ^x00000000
                                011DC P.AHA: .LONG ^x00004100, ^x00000000
                                00000000 00000000 00000000 00004002
                                00000000 00000000 00000000 00004002
                                011EC P.AHB: .LONG ^x00004002, ^x00000000, ^x00000000, ^x0000-
                                0000
                                ^x00004002, ^x00000000, ^x00000000, ^x0000-
                                0000
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 3D 011FC P.AHC: .ASCII \=DBGCVTDX\<92>\DBG$CVT_DX_DX scale fact\
20 65 6C 61 63 73 20 58 44 5F 58 44 5F 54 56 0120B
                                0121A
                                0121E
                                0122D
                                0123A P.AHD: .ASCII \BDBGCVTDX\<92>\DBG$CVT_DX_DX binary sca\
79 72 61 6E 69 62 20 58 44 5F 58 44 5F 54 56 01249
                                01258
                                0125C
                                0126B
                                0127A
                                0127D P.AHE: .ASCII \4DBGCVTDX\<92>\DBG$CVT_DX_DX: error in\
6F 72 72 65 20 20 3A 58 44 5F 58 44 5F 54 56 0128C
                                0129B
                                0129F
                                012AE
                                012B2 P.AHF: .ASCII \DBGCVTDX\<92>\DBG$CVT_DX_DX: smlint_n\
69 6C 6D 73 20 20 3A 58 44 5F 58 44 5F 54 56 012C1
                                012D0
                                012D4
                                012D7 P.AHG: .ASCII \DBGCVTDX\<92>\DBG$CVT_DX_DX: lrgint_n\
69 67 72 6C 20 20 3A 58 44 5F 58 44 5F 54 56 012E6
                                012F5
                                012F9
                                012FC P.AHH: .ASCII \DBGCVTDX\<92>\DBG$CVT_DX_DX: dec_nbds\
5F 63 65 64 20 20 3A 58 44 5F 58 44 5F 54 56 0130B
                                0131A
                                0131E
                                01320 P.AHI: .BLKB 2
                                01328 P.AHJ: .LONG ^x00004100, ^x00000000
                                01330 P.AHK: .LONG ^x00004100, ^x00000000
                                01338 P.AHL: .LONG ^x00004100, ^x00000000
                                01340 P.AHM: .LONG ^x00004220, ^x00000000
                                01348 P.AHN: .LONG ^x00004220, ^x00000000
                                01350 P.AHO: .LONG ^x00004220, ^x00000000
                                01358 P.AHP: .LONG ^x00004220, ^x00000000
                                01360 P.AHQ: .ASCII \+DBGCVTDX\<92>\DBG$CVT_DX_DX: smlfltcm\
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 2B 0136F
66 6C 6D 73 20 20 3A 58 44 5F 58 44 5F 54 56 0137E
                                01382
                                0138C P.AHR: .ASCII \plx smlint\
                                01394 P.AHS: .LONG ^x00004100, ^x00000000
                                0139C P.AHT: .LONG ^x00004100, ^x00000000
                                013A4 P.AHU: .LONG ^x00004100, ^x00000000
                                00000000 00004100
                                00000000 00004100
                                00000000 00004100
                                00000000 00004100
```

43	24	47	42	44	5C	58	44	54	56	43	47	42	44	28	013AC	P.AHV:	.LONG	^X00004220, ^X00000000	
66	6C	6D	73	20	20	3A	58	44	5F	58	44	5F	54	56	013B4	P.AHW:	.LONG	^X00004220, ^X00000000	
											6D	63	74	6C	013BC	P.AHX:	.LONG	^X00004220, ^X00000000	
															013C4	P.AHY:	.LONG	^X00004220, ^X00000000	
															013CC	P.AHZ:	.ASCII	\+DBGCVTDX\<92>\DBG\$CVT_DX_DX: smlfltcm\	
															013DB				
															013EA				
43	24	47	42	44	5C	58	44	54	56	6C	5F	78	6C	70	013EE		.ASCII	\plx_lrgint\	
6F	72	72	65	20	20	3A	58	44	5F	58	44	5F	54	56	013F8	P.AIA:	.ASCII	\4DBGCVTDX\<92>\DBG\$CVT_DX_DX: error in\	
											6E	69	20	72	01407				
72	65	76	6E	6F	63	20	65	74	2D	6F	74	2D	64	20	01416				
											6E	6F	69	73	0141A		.ASCII	\ d-to-te conversion\	
43	24	47	42	44	5C	58	44	54	56	43	47	42	44	29	01429				
66	6C	6D	73	20	20	3A	58	44	5F	58	44	5F	54	56	0142D	P.AIB:	.ASCII	\)DBGCVTDX\<92>\DBG\$CVT_DX_DX: smlfltcm\	
											6D	63	74	6C	0143C				
															0144B				
															0144F		.ASCII	\plx_nbds\	
															01457		.BLKB	1	
															01458	P.AIC:	.LONG	-	
																		^X00004002, ^X00000000, ^X00000000, ^X0000-0000	
																		-	
																		^X00004002, ^X00000000, ^X00000000, ^X0000-0000	
																		-	
																		^X00004002, ^X00000000, ^X00000000, ^X0000-0000	
																		-	
																		^X00004002, ^X00000000, ^X00000000, ^X0000-0000	
																		-	
																		^X40004004, ^X00000000, ^X00000000, ^X0000-0000	
																		-	
																		^X40004004, ^X00000000, ^X00000000, ^X0000-0000	
																		-	
																		^X40004004, ^X00000000, ^X00000000, ^X0000-0000	
																		-	
																		^X40004004, ^X00000000, ^X00000000, ^X0000-0000	
																		-	
																		^X00004002, ^X00000000, ^X00000000, ^X0000-0000	
																		-	
																		^X00004002, ^X00000000, ^X00000000, ^X0000-0000	
																		-	
																		^X00004002, ^X00000000, ^X00000000, ^X0000-0000	
																		-	
																		^X00004002, ^X00000000, ^X00000000, ^X0000-0000	
																		-	
																		^X00004002, ^X00000000, ^X00000000, ^X0000-0000	
																		-	
																		^X40004004, ^X00000000, ^X00000000, ^X0000-0000	

43 24 47 42 44  
66 67 72 6C 20

00000000	00000000	00000000	40004004	01528 P.AIP:	.LONG	- ^x40004004, ^x00000000, ^x00000000, ^x0000- 0000
00000000	00000000	00000000	40004004	01538 P.AIQ:	.LONG	- ^x40004004, ^x00000000, ^x00000000, ^x0000- 0000
00000000	00000000	00000000	40004004	01548 P.AIR:	.LONG	- ^x40004004, ^x00000000, ^x00000000, ^x0000- 0000
5C 58 44 54 56 20 3A 58 44 5F	43 47 42 44 28 58 44 5F 54 56	01558 P.AIS:	.ASCII	\DBGCVTDX\<92>\DBG\$CVT_DX_DX: lrgfltcml		
74 6E 69 6C 6D 00000000	73 6D 63 74 6C 5F 78 6C 70	01567 01576 0157A	01584 P.AIT:	.ASCII .LONG	\plx_smlint\ - ^x00004002, ^x00000000, ^x00000000, ^x0000- 0000	
00000000	00000000	00000000	00004002	01594 P.AIU:	.LONG	- ^x00004002, ^x00000000, ^x00000000, ^x0000- 0000
00000000	00000000	00000000	00004002	015A4 P.AIV:	.LONG	- ^x00004002, ^x00000000, ^x00000000, ^x0000- 0000
00000000	00000000	00000000	00004002	015B4 P.AIW:	.LONG	- ^x00004002, ^x00000000, ^x00000000, ^x0000- 0000
00000000	00000000	00000000	40004004	015C4 P.AIX:	.LONG	- ^x40004004, ^x00000000, ^x00000000, ^x0000- 0000
00000000	00000000	00000000	40004004	015D4 P.AIY:	.LONG	- ^x40004004, ^x00000000, ^x00000000, ^x0000- 0000
00000000	00000000	00000000	40004004	015E4 P.AIZ:	.LONG	- ^x40004004, ^x00000000, ^x00000000, ^x0000- 0000
00000000	00000000	00000000	40004004	015F4 P.AJA:	.LONG	- ^x40004004, ^x00000000, ^x00000000, ^x0000- 0000
00000000	00000000	00000000	00004002	01604 P.AJB:	.LONG	- ^x00004002, ^x00000000, ^x00000000, ^x0000- 0000
00000000	00000000	00000000	00004002	01614 P.AJC:	.LONG	- ^x00004002, ^x00000000, ^x00000000, ^x0000- 0000
00000000	00000000	00000000	00004002	01624 P.AJD:	.LONG	- ^x00004002, ^x00000000, ^x00000000, ^x0000- 0000
00000000	00000000	00000000	00004002	01634 P.AJE:	.LONG	- ^x00004002, ^x00000000, ^x00000000, ^x0000- 0000
00000000	00000000	00000000	40004004	01644 P.AJF:	.LONG	- ^x40004004, ^x00000000, ^x00000000, ^x0000- 0000
00000000	00000000	00000000	40004004	01654 P.AJG:	.LONG	- ^x40004004, ^x00000000, ^x00000000, ^x0000- 0000
00000000	00000000	00000000	40004004	01664 P.AJH:	.LONG	- ^x40004004, ^x00000000, ^x00000000, ^x0000-

					00000000	00000000	00000000	40004004	01674 P.AJI:	.LONG	0000							
											-	^x40004004, ^x00000000, ^x00000000, ^x0000-0000						
43	24	47	42	44	5C	58	44	54	56	43	47	42	44	2B	01684 P.AJJ:	.ASCII	\+DBGCVTDX\<92>\DBG\$CVT_DX_DX: lrgfltcml	
66	67	72	6C	20	20	3A	58	44	5F	58	44	5F	54	56	01693			
															016A2			
					74	6E	69	67	72	6C	5F	78	6C	70	016A6	.ASCII	\plx_lrgint\	
					00000000	00000000	00000000	00000000	00004002	01680 P.AJK:	.LONG	-	^x00004002, ^x00000000, ^x00000000, ^x0000-0000					
					00000000	00000000	00000000	00000000	00004002	016C0 P.AJL:	.LONG	-	^x00004002, ^x00000000, ^x00000000, ^x0000-0000					
					00000000	00000000	00000000	00000000	00004002	016D0 P.AJM:	.LONG	-	^x00004002, ^x00000000, ^x00000000, ^x0000-0000					
					00000000	00000000	00000000	00000000	00004002	016E0 P.AJN:	.LONG	-	^x00004002, ^x00000000, ^x00000000, ^x0000-0000					
					00000000	00000000	00000000	00000000	40004004	016F0 P.AJO:	.LONG	-	^x40004004, ^x00000000, ^x00000000, ^x0000-0000					
					00000000	00000000	00000000	00000000	40004004	01700 P.AJP:	.LONG	-	^x40004004, ^x00000000, ^x00000000, ^x0000-0000					
					00000000	00000000	00000000	00000000	40004004	01710 P.AJQ:	.LONG	-	^x40004004, ^x00000000, ^x00000000, ^x0000-0000					
					00000000	00000000	00000000	00000000	40004004	01720 P.AJR:	.LONG	-	^x40004004, ^x00000000, ^x00000000, ^x0000-0000					
					00000000	00000000	00000000	00000000	00004002	01730 P.AJS:	.LONG	-	^x00004002, ^x00000000, ^x00000000, ^x0000-0000					
					00000000	00000000	00000000	00000000	00004002	01740 P.AJT:	.LONG	-	^x00004002, ^x00000000, ^x00000000, ^x0000-0000					
					00000000	00000000	00000000	00000000	00004002	01750 P.AJU:	.LONG	-	^x00004002, ^x00000000, ^x00000000, ^x0000-0000					
					00000000	00000000	00000000	00000000	00004002	01760 P.AJV:	.LONG	-	^x00004002, ^x00000000, ^x00000000, ^x0000-0000					
					00000000	00000000	00000000	00000000	40004004	01770 P.AJW:	.LONG	-	^x40004004, ^x00000000, ^x00000000, ^x0000-0000					
					00000000	00000000	00000000	00000000	40004004	01780 P.AJX:	.LONG	-	^x40004004, ^x00000000, ^x00000000, ^x0000-0000					
					00000000	00000000	00000000	00000000	40004004	01790 P.AJY:	.LONG	-	^x40004004, ^x00000000, ^x00000000, ^x0000-0000					
					00000000	00000000	00000000	00000000	40004004	017A0 P.AJZ:	.LONG	-	^x40004004, ^x00000000, ^x00000000, ^x0000-0000					
43	24	47	42	44	5C	58	44	54	56	43	47	42	44	30	017B0 P.AKA:	.ASCII	\0DBGCVTDX\<92>\DBG\$CVT_DX_DX: lrgfltcml	

DBGCVTDX  
V04-000

M 10

15-Sep-1984 23:57:30

VAX-11 Bliss-32 V4.0-742

Page 106

14-Sep-1984 12:16:44

[DEBUG.SRC]DBGCVTDX.B32;1

(29)

```
66 67 72 6C 20 20 3A 58 44 5F 58 44 5F 54 56 017BF
78 6C 70 6D 63 74 6C 66 6C 6D 73 6D 63 74 6C 017CE
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 34 017D2
6F 72 72 65 20 20 3A 58 44 5F 58 44 5F 54 56 017E1 P.AKB: .ASCII \plx_smlfltcmplx\
6E 69 20 72 017FF \4DBGCVTDX\<92>\DBG$CVT_DX_DX: error in\
72 65 76 6E 6F 63 20 65 74 2D 6F 74 2D 67 20 01803 .ASCII \ g-to-te conversion\
6E 6F 69 73 01812
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 34 01816 P.AKC: .ASCII \4DBGCVTDX\<92>\DB_$CVT_DX_DX: error in\
6F 72 72 65 20 20 3A 58 44 5F 58 44 5F 54 56 01825
6E 69 20 72 01834
72 65 76 6E 6F 63 20 65 74 2D 6F 74 2D 68 20 01838 .ASCII \ h-to-te conversion\
6E 6F 69 73 01847
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 29 0184B P.AKD: .ASCII \)DBGCVTDX\<92>\DBG$CVT_DX_DX: lrgfltcm\
66 67 72 6C 20 20 3A 58 44 5F 58 44 5F 54 56 0185A
6D 63 74 6C 01869
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 23 0186D P.AKE: .ASCII \plx_nbds\
5F 63 65 64 20 20 3A 58 44 5F 58 44 5F 54 56 01875 P.AKE: .ASCII \#DBGCVTDX\<92>\DBG$CVT_DX_DX: dec_sml\
69 6C 6D 73 01884
74 6E 01893
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 23 01897 P.AKF: .ASCII \nt\
5F 63 65 64 20 20 3A 58 44 5F 58 44 5F 54 56 01899 P.AKF: .ASCII \#DBGCVTDX\<92>\DBG$CVT_DX_DX: dec_lrgi\
69 67 72 6C 018A8
74 6E 018B7
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 24 018BB P.AKG: .ASCII \nt\
73 64 62 6E 20 20 3A 58 44 5F 58 44 5F 54 56 018BD P.AKG: .ASCII \DBGCVTDX\<92>\DBG$CVT_DX_DX: nbds_sml\
6C 6D 73 5F 018CC
74 6E 69 018DB
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 24 018DF P.AKH: .ASCII \int\
73 64 62 6E 20 20 3A 58 44 5F 58 44 5F 54 56 018E2 P.AKH: .ASCII \DBGCVTDX\<92>\DBG$CVT_DX_DX: nbds_lrg\
67 72 6C 5F 018F1
74 6E 69 01900
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 29 01904 P.AKI: .ASCII \int\
73 64 62 6E 20 20 3A 58 44 5F 58 44 5F 54 56 01907 P.AKI: .ASCII \)DBGCVTDX\<92>\DBG$CVT_DX_DX: nbds_lrg\
67 72 6C 5F 01916
6E 69 20 72 01925
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 34 01929 P.AKJ: .ASCII \fltcmplx\
6F 72 72 65 20 20 3A 58 44 5F 58 44 5F 54 56 01931 P.AKJ: .ASCII \4DBGCVTDX\<92>\DBG$CVT_DX_DX: error in\
6E 69 20 72 0194F
72 65 76 6E 6F 63 20 65 74 2D 6F 74 2D 68 20 01953 .ASCII \ h-to-te conversion\
6E 6F 69 73 01962
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 22 01966 P.AKK: .ASCII \'DBGCVTDX\<92>\DBG$CVT_DX_DX: nbds_nbd\
73 64 62 6E 20 20 3A 58 44 5F 58 44 5F 54 56 01975
64 62 6E 5F 01984
43 24 47 42 44 5C 58 44 54 56 43 47 42 44 26 01988 P.AKL: .ASCII \s\
61 76 6E 69 20 20 3A 58 44 5F 58 44 5F 54 56 01989 P.AKL: .ASCII \&DBGCVTDX\<92>\DBG$CVT_DX_DX: invalid \
65 70 79 74 64 019A7
019AB .ASCII \dtype\
```

.PSECT DBG\$OWN,NOEXE, PIC,2

00008 INPUT\_STR:  
          .BLK 100  
0006C OUTPUT\_STR:  
          .BLK 100



				.PSECT	DBG\$CODE,NOWRT, SHR, PIC,0	
				.ENTRY	DBG\$CVT_DX_DX, Save R2,R3,R4,R5,R6,R7,R8,-	1763
					R9,R10,R11-	
5E	FEE8	CE	9E 00002	MOVAB	-280(SP), SP	
6D	3091	CF	DE 00007	MOVAL	661\$, (FP)	1930
59	08	AC	DO 0000C	MOVL	DESTINATION, R9	1993
5B	02	A9	9E 00010	MOVAB	2(R9), R11	
		51	D4 00014	CLRL	R1	
23		6B	91 00016	CMPB	(R11), #35	
		04	12 00019	BNEQ	1\$	
		51	D6 0001B	INCL	R1	
		0D	11 0001D	BRB	2\$	
50	04	AC	DO 0001F 1\$:	MOVL	SOURCE, R0	1994
23	02	A0	91 00023	CMPB	2(R0), #35	
		03	13 00027	BEQL	2\$	
		0091	31 00029	BRW	8\$	
0E		6B	91 0002C 2\$:	CMPB	(R11), #14	1996
		44	12 0002F	BNEQ	4\$	
50	04	AC	DO 00031	MOVL	SOURCE, R0	1997
23	02	A0	91 00035	CMPB	2(R0), #35	
		3A	12 00039	BNEQ	4\$	
58	AE 010E0017	8F	DO 0003B	MOVL	#17694743, CLASS_S_DESC	2004
5C	AE 04	A9	DO 00043	MOVL	4(R9), CLASS_S_DESC+4	2005
		7E	D4 00048	CLRL	-(SP)	2006
50	04	AC	DO 0004A	MOVL	SOURCE, R0	
	04	A0	DD 0004E	PUSHL	4(R0)	
	60	AE	9F 00051	PUSHAB	CLASS_S_DESC	
	30	AE	9F 00054	PUSHAB	TEMP	
00000000G	00	04	FB 00057	CALLS	#4, SYSSASCTIM	
	0D	50	E8 0005E	BLBS	R0, 3\$	
00000000G	00	000282A0	8F	PUSHL	#164512	2008
14	AE	01	FB 00067	CALLS	#1, LIB\$SIGNAL	
	24	AE	DO 0006E 3\$:	MOVL	TEMP, OUTPUT_STR_LEN	2009
		45	11 00073	BRB	7\$	1996
35		51	E9 00075 4\$:	BLBC	R1, 5\$	2012
50	04	AC	DO 00078	MOVL	SOURCE, R0	2013
0E	02	A0	91 0007C	CMPB	2(R0), #14	
		2B	12 00080	BNEQ	5\$	
5A	AE 010E	8F	BO 00082	MOVW	#270, CLASS_S_DESC+2	2017
50	04	AC	DC 00088	MOVL	SOURCE, R0	2018
58	AE	60	BO 0008C	MOVW	(R0), CLASS_S_DESC	
5C	AE	A0	DO 00090	MOVL	4(R0), CLASS_S_DESC+4	2019
	04	A9	DD 00095	PUSHL	4(R9)	2020
	5C	AE	9F 00098	PUSHAB	CLASS_S_DESC	
00000000G	00	02	FB 0009B	CALLS	#2, SYSSBINTIM	
	15	50	F8 000A2	BLBS	R0, 7\$	
	00028F88	8F	DD 000A5	PUSHL	#167816	2022
		06	11 000AB	BRB	6\$	
	000287D8	8F	DD 000AD 5\$:	PUSHL	#165848	2025
00000000G	00	01	FB 000B3 6\$:	CALLS	#1, LIB\$SIGNAL	
		2FD4	31 000BA 7\$:	BRW	659\$	1996
03		6C	91 000BD 8\$:	CMPB	(AP), #3	2030
		07	1B 000C0	BLEQU	9\$	

		0C	AE	10	AC	D0	000C2		MOVL	16(AP), CVT_ROUND_FLAG	2032
					03	11	000C7		BRB	10\$	2034
				0C	AE	D4	000C9	9\$:	CLRL	CVT_ROUND_FLAG	2039
		50		04	AC	D0	000CC	10\$:	MOVL	SOURCE, R0	2040
		15		02	A0	91	000D0		CMPB	2(R0), #21	2042
					12	12	000D4		BNEQ	11\$	2052
		0E			6B	91	000D6		CMPB	(R11), #14	2053
					0D	13	000D9		BEQL	11\$	2061
					50	DD	000DB		PUSHL	R0	2064
	00000000G	00			01	FB	000DD		CALLS	#1, DBG\$STRIP_ZEROES	2065
	04	AC			50	D0	000E4	11\$:	MOVL	R0, SOURCE	2068
		5A		04	AC	D0	000E8		MOVL	SOURCE, R10	2073
	04	AE		02	AA	9E	000EC		MOVAB	2(R10), 4(SP)	2074
					51	D4	000F1		CLRL	R1	2075
		0F		04	BE	91	000F3		CMPB	24(SP), #15	2076
					02	1F	000F7		BLSSU	12\$	2077
					51	D6	000F9		INCL	R1	2078
					50	D4	000FB	12\$:	CLRL	R0	2085
		15		04	BE	91	000FD		CMPB	24(SP), #21	2091
					02	1A	00101		BGTRU	13\$	2092
					50	D6	00103		INCL	R0	2093
		52			51	D2	00105	13\$:	MCOML	R1, R2	2094
	00000000'	50			52	CB	00108		BICL3	R2, R0, DECIMAL_CONVERT	2099
		0D		03	A9	91	00110		CMPB	3(R9), #13	2102
					0C	12	00114		BNEQ	14\$	2103
		34	AE	D0	AD	9E	00116		MOVAB	OUTPUT_BUFFER, OUTPUT	2110
		20	AE	04	A9	D0	0011B		MOVL	4(R9), DESTINATION_PTR	2119
					05	11	00120		BRB	15\$	2122
08	00	34	AE	04	A9	D0	00122	14\$:	MOVL	4(R9), OUTPUT	2122
		6E			00	2C	00127	15\$:	MOVCS	#0, (SP), #0, #8, SRC_INFO	2122
08	00			F8	AD		0012C				2122
					00	2C	0012E		MOVCS	#0, (SP), #0, #8, DST_INFO	2122
20	00			F0	AD		00133				2122
		6E			00	2C	00135		MOVCS	#0, (SP), #0, #32, INTMED_DATA	2122
32	20			B0	AD		0013A				2122
		6E			00	2C	0013C		MOVCS	#0, (SP), #32, #50, TEMP_BUF1	2122
32	20			FF7C	CD		00141				2122
		6E			00	2C	00144		MOVCS	#0, (SP), #32, #50, TEMP_BUF2	2122
				60	AE		00149				2122
				14	AE	D4	0014B		CLRL	OUTPUT_STR_LEN	2122
	5A	AE	010E		8F	80	0014E		MOVW	#270, CLASS_S_DESC+2	2122
	1C	AE	00000000'		EF	9E	00154		MOVAB	P.ADX, LRGST_P_LU	2122
	10	AE	00000000'		EF	9E	0015C		MOVAB	P.ADY, LRGST_D_LU	2122
	18	AE	00000000'		EF	9E	00164		MOVAB	P.ADZ, LRGST_H_LU	2122
	08	AE	00000000'		EF	9E	0016C		MOVAB	P.AEA, PACK_ZERO	2122
	F9	AD		B0	AD	9E	00174		MOVAB	INTMED_DATA, SRC_INFO+1	2122
	FD	AD			20	80	00179		MOVW	#32, SRC_INFO+5	2122
				28	AE	9F	0017D		PUSHAB	CVT_PATH	2122
				F0	AD	9F	00180		PUSHAB	DST_INFO	2122
				F8	AD	9F	00183		PUSHAB	SRC_INFO	2122
					59	DD	00186		PUSHL	R9	2122
					5A	DD	00188		PUSHL	R10	2122
	0000V	CF		05	FB	0018A			CALLS	#5, FIND_CVT_PATH	2122
		6E		50	D0	0018F			MOVL	R0, STATUS	2122
				43	18	00192			BGEQ	22\$	2119
	06	FFFFF9	8F	6E	CF	00194			CASEL	STATUS, #-7, #6	2122
001E	0016		000E	0026		0019C	16\$:		.WORD	20\$-16\$,-	2122

0016	000E	001E	001A4				
				17\$-16\$,-			
				18\$-16\$,-			
				19\$-16\$,-			
				19\$-16\$,-			
				17\$-16\$,-			
				18\$-16\$,-			
		00000000'	EF 9F 001AA 17\$:	PUSHAB	P.AEB		2124
			16 11 001B0	BRB	21\$		
		00000000'	EF 9F 001B2 18\$:	PUSHAB	P.AEC		2125
			0E 11 001B8	BRB	21\$		
		00000000'	EF 9F 001BA 19\$:	PUSHAB	P.AED		2126
			06 11 001C0	BRB	21\$		
		00000000'	EF 9F 001C2 20\$:	PUSHAB	P.AEE		2127
			01 DD 001C8 21\$:	PUSHL	#1		
		00028362	8F DD 001CA	PUSHL	#164706		
	00000000G 00		03 FB 001D0	CALLS	#3, LIB\$SIGNAL		
		00E0	8F B8 001D7 22\$:	BISPSW	#224		2136
04	FF	AD	01 E1 001DB	BBC	#1, SRC_INFO+7, 23\$		2137
			51 D4 001E0	CLRL	R1		
			04 11 001E2	BRB	24\$		
		51	AD 98 001E4 23\$:	CVTBL	SRC_INFO, R1		
04	F7	AD	01 E1 001E8 24\$:	BBC	#1, DST_INFO+7, 25\$		2138
			50 D4 001ED	CLRL	R0		
			04 11 001EF	BRB	26\$		
		50	AD 98 001F1 25\$:	CVTBL	DST_INFO, R0		
30	AE	51	50 C3 001F5 26\$:	SUBL3	R0, R1, SCALE		
06	FF	AD	01 E1 001FA	BBC	#1, SRC_INFO+7, 27\$		2139
		51	AD 98 001FF	CVTBL	SRC_INFO, R1		
			02 11 00203	BRB	28\$		
			51 D4 00205 27\$:	CLRL	R1		
		AD	01 E1 00207 28\$:	BBC	#1, DST_INFO+7, 29\$		2140
06	F7	50	AD 98 0020C	CVTBL	DST_INFO, R0		
			02 11 00210	BRB	30\$		
			50 D4 00212 29\$:	CLRL	R0		
58		51	50 C3 00214 30\$:	SUBL3	R0, R1, BIN_SCALE		
		56	AE D0 00218	MOVL	CVT_PATH, R6		2187
23		01	56 CF 0021C	CASEL	R6, #1, #35		
058B	02C7	0119	0048 00220 31\$:	.WORD	32\$-31\$,-		
0119	0CA3	118F	09FF 00228		46\$-31\$,-		
118F	0E5B	058B	02C7 00230		76\$-31\$,-		
058B	02C7	165E	1528 00238		110\$-31\$,-		
1A93	1831	17A3	0E5B 00240		159\$-31\$,-		
1E9A	0E5B	058B	1CAB 00248		277\$-31\$,-		
058B	02C7	239B	1FEE 00250		198\$-31\$,-		
2852	251A	118F	09FF 00258		46\$-31\$,-		
2AE4	0E5B	2A33	02C7 00260		76\$-31\$,-		
					110\$-31\$,-		
					227\$-31\$,-		
					277\$-31\$,-		
					332\$-31\$,-		
					344\$-31\$,-		
					76\$-31\$,-		
					110\$-31\$,-		
					227\$-31\$,-		
					357\$-31\$,-		
					368\$-31\$,-		
					390\$-31\$,-		



34	51	BE	B0	AD	01	FO	00000000'	EF	9F	002EF	38\$:	PUSHAB	P.AEF	2230
							2CD1	31	002F5			BRW	647\$	2198
							B0	AD	D0	002F8	39\$:	MOVL	INTMED_DATA, R0	
		34					BE	50	90	002FC		MOVW	R0, @OUTPUT	
		000000FF					8F	50	D1	00300		CMPL	R0, #255	
								0F	11	00307		BRB	41\$	
							B0	AD	D0	00309	40\$:	MOVL	INTMED_DATA, R0	2204
		34					BE	50	B0	0030D		MOVW	R0, @OUTPUT	
		0000FFFF					8F	50	D1	00311		CMPL	R0, #65535	
								1C	1B	00318	41\$:	BLEQU	45\$	
							26F2	31	0031A			BRW	565\$	2205
							52	F5	AD	3C	0031D	42\$:	MOVZWL	DST_INFO+5, R2
							50	01	CE	00321		MNEGL	#1, -1	2223
								0C	11	00324		BRB	44\$	2225
							01	50	EF	00326	43\$:	EXTZV	I, #1, INTMED_DATA, R1	
							50	51	F0	0032C		INSV	R1, I, #1, @OUTPUT	
							50	52	F2	00332	44\$:	AOBLSS	R2, I, 43\$	2223
							2C9F	31	00336		45\$:	BRW	649\$	2187
							02	56	D1	00339	46\$:	CMPL	R6, #2	2239
								5D	12	0033C		BNEQ	52\$	
							B0	AD	D5	0033E		TSTL	INTMED_DATA	2240
								11	18	00341		BGEQ	48\$	
							50	B0	AD	D0	00343		MOVL	INTMED_DATA, R0
								03	18	00347		BGEQ	47\$	
							50	50	CE	00349		MNEGL	R0, R0	
		B0					AD	50	D0	0034C	47\$:	MOVL	R0, INTMED_DATA	
		FF					AD	01	88	00350		BISB2	#1, SRC_INFO+7	
							30	AE	D5	00354	48\$:	TSTL	SCALE	
								0F	15	00357		BLEQ	49\$	
							50	B0	AD	9E	00359		MOVAB	INTMED_DATA, R0
							00000000G	00	16	0035D		JSB	LIB\$SCVT_SCALE_OU_UP_BY_10_R1	
							30	AE	D7	00363		DECL	SCALE	
								EC	11	00366		BRB	48\$	
								0F	18	00368	49\$:	BGEQ	50\$	
							50	B0	AD	9E	0036A		MOVAB	INTMED_DATA, R0
							00000000G	00	16	0036E		JSB	LIB\$SCVT_SCALE_OU_DOWN_BY_10_R1	
							30	AE	D6	00374		INCL	SCALE	

		EF	11	00377	BRB	49\$				
		58	D5	00379	50\$: TSTL	BIN_SCALE				
		0E	15	0037B	BLEQ	51\$				
50	B0	AD	9E	0037D	MOVAB	INTMED_DATA, R0				
	00000000G	00	16	00381	JSB	DBG\$CVT_SCALE_OU_UP_BY_2_R1				
		58	D7	00387	DECL	BIN_SCALE				
		EE	11	00389	BRB	50\$				
		5A	18	0038B	51\$: BGEQ	57\$				
50	B0	AD	9E	0038D	MOVAB	INTMED_DATA, R0				
	00000000G	00	16	00391	JSB	DBG\$CVT_SCALE_OU_DOWN_BY_2_R1				
		58	D6	00397	INCL	BIN_SCALE				
		FO	11	00399	BRB	51\$				
08		56	D1	0039B	52\$: CMPL	R6, #8		2244		
		47	12	0039E	BNEQ	57\$				
	30	AE	D5	003A0	53\$: TSTL	SCALE		2245		
		0F	15	003A3	BLEQ	54\$				
50	B0	AD	9E	003A5	MOVAB	INTMED_DATA, R0				
	00000000G	00	16	003A9	JSB	LIB\$CVT_SCALE_OU_UP_BY_10_R1				
	30	AE	D7	003AF	DECL	SCALE				
		EC	11	003B2	BRB	53\$				
		0F	18	003B4	54\$: BGEQ	55\$				
50	B0	AD	9E	003B6	MOVAB	INTMED_DATA, R0				
	00000000G	00	16	003BA	JSB	LIB\$CVT_SCALE_OU_DOWN_BY_10_R1				
	30	AE	D6	003C0	INCL	SCALE				
		EF	11	003C3	BRB	54\$				
		58	D5	003C5	55\$: TSTL	BIN_SCALE				
		0E	15	003C7	BLEQ	56\$				
50	B0	AD	9E	003C9	MOVAB	INTMED_DATA, R0				
	00000000G	00	16	003CD	JSB	DBG\$CVT_SCALE_OU_UP_BY_2_R1				
		58	D7	003D3	DECL	BIN_SCALE				
		EE	11	003D5	BRB	55\$				
		0E	18	003D7	56\$: BGEQ	57\$				
50	B0	AD	9E	003D9	MOVAB	INTMED_DATA, R0				
	00000000G	00	16	003DD	JSB	DBG\$CVT_SCALE_OU_DOWN_BY_2_R1				
		58	D6	003E3	INCL	BIN_SCALE				
		FO	11	003E5	BRB	56\$				
		6B	91	003E7	57\$: CMPB	(R11), #4		2253		
		31	12	003EA	BNEQ	59\$				
50	B4	AD	B8	AD	C9	003EC	BISL3	INTMED_DATA+8, INTMED_DATA+4, R0	2255	
		50	BC	AD	C8	003F2	BISL2	INTMED_DATA+12, R0		
				15	13	003F6	BEQL	58\$		
				00000000G	00	DD	003F8	PUSHL	DBG\$GL_OPCODE_NAME	2256
					01	DD	003FE	PUSHL	#1	
				000286A3	8F	DD	00400	PUSHL	#165539	
	00000000G	00	03	FB	00406	CALLS	#3, LIB\$SIGNAL			
	34	BE	B0	AD	D0	0040D	58\$: MOVL	INTMED_DATA, @OUTPUT	2257	
		60	FF	AD	E9	00412	BLBC	SRC_INFO+7, 64\$	2258	
	34	BE	34	BE	CE	00416	MNEGL	@OUTPUT, @OUTPUT	2260	
				59	11	0041B	BRB	64\$	2250	
		05		6B	91	0041D	59\$: CMPB	(R11), #5	2263	
				05	13	00420	BEQL	60\$		
		09		6B	91	00422	CMPB	(R11), #9		
				51	12	00425	BNEQ	65\$		
50	B8	AD	BC	AD	C9	00427	60\$: BISL3	INTMED_DATA+12, INTMED_DATA+8, R0	2265	
				15	13	0042D	BEQL	61\$		
				00000000G	00	DD	0042F	PUSHL	DBG\$GL_OPCODE_NAME	2266
					01	DD	00435	PUSHL	#1	

00000000G	00	000286A3	8F	DD	00437	PUSHL	#165539		
	26		03	FB	0043D	CALLS	#3, LIBSSIGNAL		
		FF	AD	E9	00444	61\$: BLBC	SRC INFO+7, 63\$	2267	
		B0	AD	D5	00448	TSTL	INTMED_DATA	2269	
			14	12	0044B	BNEQ	62\$		
80000000	8F	B4	AD	D1	0044D	CMPL	INTMED_DATA+4, #-2147483648	2272	
			17	13	00455	BEQL	63\$		
B4	AD	B4	AD	D2	00457	MCOML	INTMED_DATA+4, INTMED_DATA+4	2275	
		B4	AD	D6	0045C	INCL	INTMED_DATA+4	2276	
			0D	11	0045F	BRB	63\$	2269	
B0	AD	B0	AD	D2	00461	62\$: MCOML	INTMED_DATA, INTMED_DATA	2281	
B4	AD	B4	AD	D2	00466	MCOML	INTMED_DATA+4, INTMED_DATA+4	2282	
		B0	AD	D6	0046B	INCL	INTMED_DATA	2283	
	50	34	AE	D0	0046E	63\$: MOVL	OUTPUT, R0	2285	
	60	B0	AD	7D	00472	MOVQ	INTMED_DATA, (R0)		
			51	11	00476	64\$: BRB	72\$	2250	
	1A		6B	91	00478	65\$: CMPB	(R11), #26	2289	
			4F	12	0047B	BNEQ	73\$		
	42	FF	AD	E9	0047D	BLBC	SRC INFO+7, 71\$	2294	
		B0	AD	D5	00481	TSTL	INTMED_DATA	2302	
			14	12	00484	BNEQ	66\$		
		B4	AD	D5	00486	TSTL	INTMED_DATA+4	2303	
			0F	12	00489	BNEQ	66\$		
		B8	AD	D5	0048B	TSTL	INTMED_DATA+8	2304	
			0A	12	0048E	BNEQ	66\$		
80000000	8F	BC	AD	D1	00490	CMPL	INTMED_DATA+12, #-2147483648	2305	
			29	13	00498	BEQL	71\$		
			50	D4	0049A	66\$: CLRL	NEXT LONGWORD	2313	
B0	AD40	B0	AD40	D2	0049C	67\$: MCOML	INTMED_DATA[NEXT_LONGWORD], INTMED_DATA-	2315	
							[NEXT_LONGWORD]		
F5	50		03	F3	004A3	AOBLEQ	#3, NEXT LONGWORD, 67\$	2314	
			50	D4	004A7	CLRL	NEXT LONGWORD	2319	
	51	B0	AD40	DE	004A9	68\$: MOVAL	INTMED_DATA[NEXT_LONGWORD], R1	2320	
FFFFFFFF	8F		61	D1	004AE	CMPL	(R1), #-1		
			04	12	004B5	BNEQ	69\$		
			61	D4	004B7	CLRL	(R1)	2322	
			04	11	004B9	BRB	70\$		
			61	D6	004BB	69\$: INCL	(R1)	2326	
			04	11	004BD	BRB	71\$	2324	
			03	F3	004BF	70\$: AOBLEQ	#3, NEXT LONGWORD, 68\$	2320	
34	E6	B0	AD	10	28	004C3	71\$: MOVCS	#16, INTMED_DATA, @OUTPUT	2332
	BE		2B0C	31	004C9	72\$: BRW	649\$	2250	
			02	56	D1	004CC	73\$: CMPL	R6, #2	2338
				08	12	004CF	BNEQ	74\$	
		00000000'	EF	9F	004D1	PUSHAB	P.AEG	2339	
			0B	11	004D7	BRB	75\$		
			56	D1	004D9	74\$: CMPL	R6, #8	2341	
			EB	12	004DC	BNEQ	72\$		
		00000000'	EF	9F	004DE	PUSHAB	P.AEH	2342	
			2AE2	31	004E4	75\$: BRW	647\$		
			56	D1	004E7	76\$: CMPL	R6, #3	2352	
			68	12	004EA	BNEQ	81\$		
B0	AD	B0	AD	6E	004EC	CVILD	INTMED_DATA, INTMED_DATA	2353	
			58	D5	004F1	77\$: TSTL	BIN_SCALE		
			15	15	004F3	BLEQ	78\$		
			51	B0	AD	9E	004F5		
		50	00000000'	EF	9E	004F9	MOVAB	INTMED_DATA, R1	
							MOVAB	P.AEI, -R0	

		00000000G	00	16	00500	JSB	DBG\$CVT_MULD2_R1	
			58	D7	00506	DECL	BIN_SCALE	
			E7	11	00508	BRB	77\$	
			15	18	0050A	78\$:	BGEQ	79\$
51		B0	AD	9E	0050C	MOVAB	INTMED_DATA, R1	
50		00000000'	EF	9E	00510	MOVAB	P.AEJ, -R0	
		0' 00000G	00	16	00517	JSB	DBG\$CVT_DIVD2_R1	
			58	D6	0051D	INCL	BIN_SCALE	
			E9	11	0051F	BRB	78\$	
		30	AE	D5	00521	79\$:	TSTL	SCALE
			16	15	00524	BLEQ	80\$	
51		B0	AD	9E	00526	MOVAB	INTMED_DATA, R1	
50		00000000'	EF	9E	0052A	MOVAB	P.AEK, -R0	
		00000000G	00	16	00531	JSB	DBG\$CVT_MULD2_R1	
		30	AE	D7	00537	DECL	SCALE	
			E5	11	0053A	BRB	79\$	
			7C	18	0053C	80\$:	BGEQ	85\$
51		B0	AD	9E	0053E	MOVAB	INTMED_DATA, R1	
50		00000000'	EF	9E	00542	MOVAB	P.AEL, -R0	
		00000000G	00	16	00549	JSB	DBG\$CVT_DIVD2_R1	
		30	AE	D6	0054F	INCL	SCALE	
			E8	11	00552	BRB	80\$	
09			56	D1	00554	81\$:	CMPL	R6, #9
			7C	12	00557	BNEQ	87\$	
51		FF7C	CD	9E	00559	MOVAB	TEMP_BUF1, R1	
50		B0	AD	9E	0055E	MOVAB	INTMED_DATA, R0	
		00000000G	00	16	00562	JSB	DBG\$CVT_CVTR0UD_R1	
	B0	AD	FF7C	CD	08	28	00568	MOV C3
			58	D5	0056F	82\$:	TSTL	BIN_SCALE
			15	15	00571	BLEQ	83\$	
51		B0	AD	9E	00573	MOVAB	INTMED_DATA, R1	
50		00000000'	EF	9E	00577	MOVAB	P.AEM, -R0	
		00000000G	00	16	0057E	JSB	DBG\$CVT_MULD2_R1	
			58	D7	00584	DECL	BIN_SCALE	
			E7	11	00586	BRB	82\$	
			15	18	00588	83\$:	BGEQ	84\$
51		B0	AD	9E	0058A	MOVAB	INTMED_DATA, R1	
50		00000000'	EF	9E	0058E	MOVAB	P.AEN, -R0	
		00000000G	00	16	00595	JSB	DBG\$CVT_DIVD2_R1	
			58	D6	0059B	INCL	BIN_SCALE	
			E9	11	0059D	BRB	83\$	
		30	AE	D5	0059F	84\$:	TSTL	SCALE
			16	15	005A2	BLEQ	85\$	
51		B0	AD	9E	005A4	MOVAB	INTMED_DATA, R1	
50		00000000'	EF	9E	005A8	MOVAB	P.AEO, -R0	
		00000000G	00	16	005AF	JSB	DBG\$CVT_MULD2_R1	
		30	AE	D7	005B5	DECL	SCALE	
			E5	11	005B8	BRB	84\$	
			03	19	005BA	85\$:	BLSS	86\$
			0179	31	005BC	BRW	98\$	
51		B0	AD	9E	005BF	86\$:	MOVAB	INTMED_DATA, R1
50		00000000'	EF	9E	005C3	MOVAB	P.AEP, -R0	
		00000000G	00	16	005CA	JSB	DBG\$CVT_DIVD2_R1	
		30	AE	D6	005D0	INCL	SCALE	
			E5	11	005D3	BRB	85\$	
0F			56	D1	005D5	87\$:	CMPL	R6, #15
			03	13	005D8	BEQL	88\$	

2357

2358

2362



				00AA	31	005DA	BRW	93\$	
				58	D5	005DD	88\$: TSTL	BIN_SCALE	2363
				26	15	005DF	BLEQ	89\$	
51		B0		AD	9E	005E1	MOVAB	INTMED_DATA, R1	
50	00000000	'		EF	9E	005E5	MOVAB	P.AEQ, R0	
	00000000G			00	16	005EC	JSB	DBG\$CVT_MULD2_R1	
51		B8		AD	9E	005F2	MOVAB	INTMED_DATA+8, R1	
50	00000000	'		EF	9E	005F6	MOVAB	P.AER, R0	
	00000000G			00	16	005FD	JSB	DBG\$CVT_MULD2_R1	
				58	D7	00603	DECL	BIN_SCALE	
				D6	11	00605	BRB	88\$	
				26	18	00607	89\$: BGEQ	90\$	
51		B0		AD	9E	00609	MOVAB	INTMED_DATA, R1	
50	00000000	'		EF	9E	0060D	MOVAB	P.AES, R0	
	00000000G			00	16	00614	JSB	DBG\$CVT_DIVD2_R1	
51		B8		AD	9E	0061A	MOVAB	INTMED_DATA+8, R1	
50	00000000	'		EF	9E	0061E	MOVAB	P.AET, R0	
	00000000G			00	16	00625	JSB	DBG\$CVT_DIVD2_R1	
				58	D6	0062B	INCL	BIN_SCALE	
				D8	11	0062D	BRB	89\$	
				30	AE	D5	90\$: TSTL	SCALE	
				27	15	00632	BLEQ	91\$	
51		B0		AD	9E	00634	MOVAB	INTMED_DATA, R1	
50	00000000	'		EF	9E	00638	MOVAB	P.AEU, R0	
	00000000G			00	16	0063F	JSB	DBG\$CVT_MULD2_R1	
51		B8		AD	9E	00645	MOVAB	INTMED_DATA+8, R1	
50	00000000	'		EF	9E	00649	MOVAB	P.AEV, R0	
	00000000G			00	16	00650	JSB	DBG\$CVT_MULD2_R1	
				30	AE	D7	DECL	SCALE	
				D4	11	00659	BRB	90\$	
				03	19	0065B	91\$: BLSS	92\$	
				00D8	31	0065D	BRW	98\$	
51		B0		AD	9E	00660	92\$: MOVAB	INTMED_DATA, R1	
50	00000000	'		EF	9E	00664	MOVAB	P.AEW, R0	
	00000000G			00	16	0066B	JSB	DBG\$CVT_DIVD2_R1	
51		B8		AD	9E	00671	MOVAB	INTMED_DATA+8, R1	
50	00000000	'		EF	9E	00675	MOVAB	P.AEX, R0	
	00000000G			00	16	0067C	JSB	DBG\$CVT_DIVD2_R1	
				30	AE	D6	INCL	SCALE	
				D4	11	00685	BRB	91\$	
				56	D1	00687	93\$: CMPL	R6, #27	2367
				6B	12	0068A	BNEQ	96\$	
2C	AE	FD		AD	3C	0068C	MOVZWL	SRC_INFO+5, NO_DIGITS	2368
09		03		AA	91	00691	CMPB	3(RT0), #9	
				0A	12	00695	BNEQ	94\$	
30	AE	08		AA	98	00697	CVTBL	8(R10), SCALE	
30	AE	30		AE	CE	0069C	MNEGL	SCALE, SCALE	
FF7C	CD	B0	AD	2C	AE	08	94\$: CVTPS	NO DIGITS, INTMED_DATA, NO_DIGITS, -	
								TEMP_BUF1	
58	AE	2C	AE	01	A1	006AB	ADDW3	#1, NO DIGITS, CLASS_S_DESC	
		5C	AE	FF7C	CD	9E	MOVAB	TEMP_BUF1, CLASS_S_DESC+4	
			7E	44	8F	9A	MOVZBL	#68, -(SP)	
				34	AE	DD	PUSHL	SCALE	
				7E	D4	006BE	CLRL	-(SP)	
				B0	AD	9F	PUSHAB	INTMED_DATA	
				68	AE	9F	PUSHAB	CLASS_S_DESC	
00000000G	00			05	FB	006C6	CALLS	#5, OTS\$CVT_T_D	

	6E		50	D0	006CD	MOVL	R0, STATUS		
	65		6E	E8	006D0	BLBS	STATUS, 98\$		
	50	00000000G	00	D0	006D3	MOVL	DBG\$GL_OPCODE_NAME, R0		
		30	AE	D5	006DA	TSTL	SCALE		
			0C	18	006DD	BGEQ	95\$		
			50	DD	006DF	PUSHL	R0		
		0002869B	01	DD	006E1	PUSHL	#1		
			8F	DD	006E3	PUSHL	#165531		
			46	11	006E9	BRB	97\$		
			50	DD	006EB	95\$: PUSHL	R0		
			01	DD	006ED	PUSHL	#1		
		00028A02	8F	DD	006EF	PUSHL	#166402		
			3A	11	006F5	BRB	97\$		
	21		56	D1	006F7	96\$: CMPL	R6, #33		2372
			3C	12	006FA	BNEQ	98\$		
58	AE	FD	AD	B0	006FC	MOVW	SRC_INFO+5, CLASS_S_DESC		2374
5C	AE	F9	AD	D0	00701	MOVL	SRC_INFO+1, CLASS_S_DESC+4		2375
	7E	55	8F	9A	00706	MOVZBL	#85, -(SP)		2377
	7E	34	AE	CE	0070A	MNEGL	SCALE, -(SP)		2376
			7E	D4	0070E	CLRL	-(SP)		
		B0	AD	9F	00710	PUSHAB	INTMED_DATA		
		68	AE	9F	00713	PUSHAB	CLASS_S_DESC		
	00000000G	00	05	FB	00716	CALLS	#5, OTS\$CVT_T_D		
		6E	50	D0	0071D	MOVL	R0, STATUS		
		15	6E	E8	00720	BLBS	STATUS, 98\$		2378
			00	DD	00723	PUSHL	DBG\$GL_OPCODE_NAME		
		00000000G	01	DD	00729	PUSHL	#1		
			8F	DD	0072B	PUSHL	#164504		
	00000000G	00	03	FB	00731	97\$: CALLS	#3, LIB\$SIGNAL		
01		0A	6B	8F	00738	98\$: CASEB	(R11), #10, #1		2382
		0064	005D		0073C	99\$: .WORD	107\$-99\$,-		
							108\$-99\$		
01		0C	6B	8F	00740	CASEB	(R11), #12, #1		2395
		04C8	0046		00744	100\$: .WORD	106\$-100\$,-		
							156\$-100\$		
		03	56	D1	00748	CMPL	R6, #3		2410
			08	12	0074B	BNEQ	101\$		
		00000000'	EF	9F	0074D	PUSHAB	P.AEY		2411
			32	11	00753	BRB	105\$		
		09	56	D1	00755	101\$: CMPL	R6, #9		2412
			08	12	00758	BNEQ	102\$		
		00000000'	EF	9F	0075A	PUSHAB	P.AEZ		2413
			25	11	00760	BRB	105\$		
		0F	56	D1	00762	102\$: CMPL	R6, #15		2414
			08	12	00765	BNEQ	103\$		
		00000000'	EF	9F	00767	PUSHAB	P.AFA		2415
			18	11	0076D	BRB	105\$		
		1B	56	D1	0076F	103\$: CMPL	R6, #27		2416
			08	12	00772	BNEQ	104\$		
		00000000'	EF	9F	00774	PUSHAB	P.AFB		2417
			0B	11	0077A	BRB	105\$		
		21	56	D1	0077C	104\$: CMPL	R6, #33		2418
			27	12	0077F	BNEQ	109\$		
		00000000'	EF	9F	00781	PUSHAB	P.AFC		2419
			0440	31	00787	105\$: BRW	151\$		
		50	34	AE	D0	0078A	106\$: MOVL	OUTPUT, R0	2400
		60	B0	AD	76	0078E	CVTDF	INTMED_DATA, (R0)	

04	A0	B8	AD	76	00792	CVTDF	INTMED_DATA+8, 4(R0)	2401
			OF	11	00797	BRB	109\$	2395
34	BE	B0	AD	76	00799	107\$:	CVTDF	INTMED_DATA, @OUTPUT
			08	11	0079E	BRB	109\$	2386
50		34	AE	D0	007A0	108\$:	MOVL	OUTPUT, R0
60		B0	AD	7D	007A4		MOVQ	INTMED_DATA, (R0)
			0467	31	007A8	109\$:	BRW	157\$
04			56	D1	007AB	110\$:	CMPL	R6, #4
			71	12	007AE		BNEQ	115\$
51		B0	AD	9E	007B0		MOVAB	INTMED_DATA, R1
50		B0	AD	9E	007B4		MOVAB	INTMED_DATA, R0
		00000000G	00	16	007B8		JSB	DBGSCVT_CVTLM_R1
			58	D5	007BE	111\$:	TSTL	BIN_SCALE
			15	15	007C0		BLEQ	112\$
51		B0	AD	9E	007C2		MOVAB	INTMED_DATA, R1
50		00000000'	EF	9E	007C6		MOVAB	P.AFD, -R0
		00000000G	00	16	007CD		JSB	DBGSCVT_MULH2_R1
			58	D7	007D3		DECL	BIN_SCALE
			E7	11	007D5		BRB	111\$
			15	18	007D7	112\$:	BGEQ	113\$
51		B0	AD	9E	007D9		MOVAB	INTMED_DATA, R1
50		00000000'	EF	9E	007DD		MOVAB	P.AFE, -R0
		00000000G	00	16	007E4		JSB	DBGSCVT_DIVH2_R1
			58	D6	007EA		INCL	BIN_SCALE
			E9	11	007EC		BRB	112\$
		30	AE	D5	007EE	113\$:	TSTL	SCALE
			16	15	007F1		BLEQ	114\$
51		B0	AD	9E	007F3		MOVAB	INTMED_DATA, R1
50		00000000'	EF	9E	007F7		MOVAB	P.AFF, -R0
		00000000G	00	16	007FE		JSB	DBGSCVT_MULH2_R1
		30	AE	D7	00804		DECL	SCALE
			E5	11	00807		BRB	113\$
			7C	18	00809	114\$:	BGEQ	119\$
51		B0	AD	9E	0080B		MOVAB	INTMED_DATA, R1
50		00000000'	EF	9E	0080F		MOVAB	P.AFG, -R0
		00000000G	00	16	00816		JSB	DBGSCVT_DIVH2_R1
		30	AE	D6	0081C		INCL	SCALE
			E8	11	0081F		BRB	114\$
0A			56	D1	00821	115\$:	CMPL	R6, #10
			7C	12	00824		BNEQ	121\$
51		FF7C	CD	9E	00826		MOVAB	TEMP_BUF1, R1
50		B0	AD	9E	0082B		MOVAB	INTMED_DATA, R0
		00000000G	00	16	0082F		JSB	DBGSCVT_CVTROUH_R1
			10	28	00835		MOVC3	#16, TEMP_BUF1, -INTMED_DATA
			58	D5	0083C	116\$:	TSTL	BIN_SCALE
			15	15	0083E		BLEQ	117\$
51		B0	AD	9E	00840		MOVAB	INTMED_DATA, R1
50		00000000'	EF	9E	00844		MOVAB	P.AFH, -R0
		00000000G	00	16	0084B		JSB	DBGSCVT_MULH2_R1
			58	D7	00851		DECL	BIN_SCALE
			E7	11	00853		BRB	116\$
			15	18	00855	117\$:	BGEQ	118\$
51		B0	AD	9E	00857		MOVAB	INTMED_DATA, R1
50		00000000'	EF	9E	0085B		MOVAB	P.AFI, -R0
		00000000G	00	16	00862		JSB	DBGSCVT_DIVH2_R1
			58	D6	00868		INCL	BIN_SCALE
			E9	11	0086A		BRB	117\$

2437  
2438

```

      30 AE D5 0086C 118$: TSTL SCALE
      16 15 0086F BLEQ 119$
51      B0 AD 9E 00871 MOVAB INTMED_DATA, R1
50 00000000' EF 9E 00875 MOVAB P.AFJ, -R0
      00 16 0087C JSB DBG$CVT_MULH2_R1
      30 AE D7 00882 DECL SCALE
      E5 11 00885 BRB 118$
      03 19 00887 119$: BLSS 120$
      02EF 31 00889 BRW 144$
51      B0 AD 9E 0088C 120$: MOVAB INTMED_DATA, R1
50 00000000' EF 9E 00890 MOVAB P.AFK, -R0
      00 16 00897 JSB DBG$CVT_DIVH2_R1
      30 AE D6 0089D INCL SCALE
      E5 11 008A0 BRB 119$
10      56 D1 008A2 121$: CMPL R6, #16
      03 13 008A5 BEQL 122$
      00CE 31 008A7 BRW 128$
51      FF7C CD 9E 008AA 122$: MOVAB TEMP_BUF1, R1
50      B0 AD 9E 008AF MOVAB INTMED_DATA, R0
      00000000G 00 16 008B3 JSB DBG$CVT_CVTDH_R1
51      C0 AD 9E 008B9 MOVAB INTMED_DATA+16, R1
50      B8 AD 9E 008BD MOVAB INTMED_DATA+8, R0
      00000000G 00 16 008C1 JSB DBG$CVT_CVTDH_R1
      B0 AD FF7C CD 10 28 008C7 MOV3 #16, TEMP_BUF1, INTMED_DATA
      58 D5 008CE 123$: TSTL BIN_SCALE
      26 15 008D0 BLEQ 124$
51      B0 AD 9E 008D2 MOVAB INTMED_DATA, R1
50 00000000' EF 9E 008D6 MOVAB P.AFL, -R0
      00000000G 00 16 008DD JSB DBG$CVT_MULH2_R1
51      C0 AD 9E 008E3 MOVAB INTMED_DATA+16, R1
50 00000000' EF 9E 008E7 MOVAB P.AFM, -R0
      00000000G 00 16 008EE JSB DBG$CVT_MULH2_R1
      58 D7 008F4 DECL BIN_SCALE
      D6 11 008F6 BRB 123$
      26 18 008F8 124$: BGEQ 125$
51      B0 AD 9E 008FA MOVAB INTMED_DATA, R1
50 00000000' EF 9E 008FE MOVAB P.AFN, -R0
      00000000G 00 16 00905 JSB DBG$CVT_DIVH2_R1
51      C0 AD 9E 0090B MOVAB INTMED_DATA+16, R1
50 00000000' EF 9E 0090F MOVAB P.AFO, -R0
      00000000G 00 16 00916 JSB DBG$CVT_DIVH2_R1
      58 D6 0091C INCL BIN_SCALE
      D8 11 0091E BRB 124$
      30 AE D5 00920 125$: TSTL SCALE
      27 15 00923 BLEQ 126$
51      B0 AD 9E 00925 MOVAB INTMED_DATA, R1
50 00000000' EF 9E 00929 MOVAB P.AFP, -R0
      00000000G 00 16 00930 JSB DBG$CVT_MULH2_R1
51      C0 AD 9E 00936 MOVAB INTMED_DATA+16, R1
50 00000000' EF 9E 0093A MOVAB P.AFO, -R0
      00000000G 00 16 00941 JSB DBG$CVT_MULH2_R1
      30 AE D7 00947 DECL SCALE
      D4 11 0094A BRB 125$
      03 19 0094C 126$: BLSS 127$
      022A 31 0094E BRW 144$
51      B0 AD 9E 00951 127$: MOVAB INTMED_DATA, R1
50 00000000' EF 9E 00955 MOVAB P.AFR, -R0
```

2442

2443

			00000000G	00	16	0095C	JSB	DBG\$CVT_DIVH2_R1	
51			C0	AD	9E	00962	MOVAB	INTMED_DATA+16, R1	
50			00000000'	EF	9E	00966	MOVAB	P.AFS, -R0	
			00000000G	00	16	0096D	JSB	DBG\$CVT_DIVH2_R1	
			30	AE	D6	00973	INCL	SCALE	
				D4	11	00976	BRB	126\$	
16				56	D1	00978	128\$:	CMPL	R6, #22
				03	13	0097B	BEQL	129\$	2447
				0186	31	0097D	BRJ	140\$	
1B	04			BE	91	00980	129\$:	CMPB	@4(SP), #27
				09	13	00984	BEQL	130\$	2449
1D	04			BE	91	00986	CMPB	@4(SP), #29	2450
				03	13	0098A	BEQL	130\$	
				00CB	31	0098C	BRW	135\$	
51		FF7C		CD	9E	0098F	130\$:	MOVAB	TEMP_BUF1, R1
50		B0		AD	9E	00994	MOVAB	INTMED_DATA, R0	2451
			00000000G	00	16	00998	JSB	DBG\$CVT_CVTGH_R1	
51			C0	AD	9E	0099E	MOVAB	INTMED_DATA+16, R1	
50			B8	AD	9E	009A2	MOVAB	INTMED_DATA+8, R0	
			00000000G	00	16	009A6	JSB	DBG\$CVT_CVTGH_R1	
B0	AD	FF7C	CD		10	28	009AC	MOV3	#16, TEMP_BUF1, INTMED_DATA
				58	D5	009B3	131\$:	TSTL	BIN_SCALE
				26	15	009B5	BLEQ	132\$	
51		B0		AD	9E	009B7	MOVAB	INTMED_DATA, R1	
50			00000000'	EF	9E	009BB	MOVAB	P.AFT, -R0	
			00000000G	00	16	009C2	JSB	DBG\$CVT_MULH2_R1	
51			C0	AD	9E	009C8	MOVAB	INTMED_DATA+16, R1	
50			00000000'	EF	9E	009CC	MOVAB	P.AFU, -R0	
			00000000G	00	16	009D3	JSB	DBG\$CVT_MULH2_R1	
				58	D7	009D9	DECL	BIN_SCALE	
				D6	11	009DB	BRB	131\$	
				26	18	009DD	132\$:	BGEQ	133\$
51		B0		AD	9E	009DF	MOVAB	INTMED_DATA, R1	
50			00000000'	EF	9E	009E3	MOVAB	P.AFV, -R0	
			00000000G	00	16	009EA	JSB	DBG\$CVT_DIVH2_R1	
51			C0	AD	9E	009F0	MOVAB	INTMED_DATA+16, R1	
50			00000000'	EF	9E	009F4	MOVAB	P.AFW, -R0	
			00000000G	00	16	009FB	JSB	DBG\$CVT_DIVH2_R1	
				58	D6	00A01	INCL	BIN_SCALE	
				D8	11	00A03	BRB	132\$	
			30	AE	D5	00A05	133\$:	TSTL	SCALE
				27	15	00A08	BLEQ	134\$	
51		B0		AD	9E	00A0A	MOVAB	INTMED_DATA, R1	
50			00000000'	EF	9E	00A0E	MOVAB	P.AFX, -R0	
			00000000G	00	16	00A15	JSB	DBG\$CVT_MULH2_R1	
51			C0	AD	9E	00A1B	MOVAB	INTMED_DATA+16, R1	
50			00000000'	EF	9E	00A1F	MOVAB	P.AFY, -R0	
			00000000G	00	16	00A26	JSB	DBG\$CVT_MULH2_R1	
			30	AE	D7	00A2C	DECL	SCALE	
				D4	11	00A2F	BRB	133\$	
				51	18	00A31	134\$:	BGEQ	136\$
51		B0		AD	9E	00A33	MOVAB	INTMED_DATA, R1	
50			00000000'	EF	9E	00A37	MOVAB	P.AFZ, -R0	
			00000000G	00	16	00A3E	JSB	DBG\$CVT_DIVH2_R1	
51			C0	AD	9E	00A44	MOVAB	INTMED_DATA+16, R1	
50			00000000'	EF	9E	00A48	MOVAB	P.AGA, -R0	
			00000000G	00	16	00A4F	JSB	DBG\$CVT_DIVH2_R1	

PC	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419	Op420	Op421	Op422	Op423	Op424	Op425	Op426	Op427	Op428	Op429	Op430	Op431	Op432	Op433	Op434	Op435	Op436	Op437	Op438	Op439	Op440	Op441	Op442	Op443	Op444	Op445	Op446	Op447	Op448	Op449	Op450	Op451	Op452	Op453	Op454	Op455	Op456	Op457	Op458	Op459	Op460	Op461	Op462	Op463	Op464	Op465	
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--

			68	AE 9F 00B42	PUSHAB	CLASS S DESC	
	00000000G	00	05	FB 00B45	CALLS	#5, OTSCVT-T-H	
		6E	50	DO 00B4C	MOVL	R0, STATUS	
		29	6E	E8 00B4F	BLBS	STATUS, 144\$	
		50	00000000G	00	DO 00B52	MOVL	DBG\$GL_OPCODE_NAME, R0
			30	AE D5 00B57	TSTL	SCALE	
				OC 18 00B5C	RGEQ	142\$	
				50	DD 00B5E	PUSHL	R0
				01	DD 00B60	PUSHL	#1
			0002869B	8F	DD 00B62	PUSHL	#165531
				0A	11 00B68	BRB	143\$
				50	DD 00B6A	142\$:	PUSHL R0
				01	DD 00B6C	PUSHL	#1
			00028A02	8F	DD 00B6E	PUSHL	#166402
	00000000G	00	03	FB 00B74	143\$:	CALLS	#3, LIB\$SIGNAL
01		1B	6B	8F 00B7B	144\$:	CASEB	(R11), #27, #1
		008D	007D	00B7F	145\$:	.WORD	154\$-145\$,-
							156\$-145\$
01		1D	6B	8F 00B83	CASEB	(R11), #29, #1	
		006D	0054	00B87	146\$:	.WORD	152\$-146\$,-
							153\$-146\$
			04	56	D1 00B8B	CMPL	R6, #4
				08	12 00B8E	BNEQ	147\$
			00000000'	EF	9F 00B90	PUSHAB	P.AGJ
				32	11 00B96	BRB	151\$
			0A	56	D1 00B98	147\$:	CMPL R6, #10
				08	12 00B9B	BNEQ	148\$
			00000000'	EF	9F 00B9D	PUSHAB	P.AGK
				25	11 00BA3	BRB	151\$
			10	56	D1 00BA5	148\$:	CMPL R6, #16
				08	12 00BA8	BNEQ	149\$
			00000000'	EF	9F 00BAA	PUSHAB	P.AGL
				18	11 00BB0	BRB	151\$
			16	56	D1 00BB2	149\$:	CMPL R6, #22
				08	12 00BB5	BNEQ	150\$
			00000000'	EF	9F 00BB7	PUSHAB	P.AGM
				0B	11 00BBD	BRB	151\$
			1C	56	D1 00BBF	150\$:	CMPL R6, #28
				4E	12 00BC2	BNEQ	157\$
			00000000'	EF	9F 00BC4	PUSHAB	P.AGN
				01	DD 00BCA	151\$:	PUSHL #1
			00028362	8F	DD 00BCC	PUSHL	#164706
	00000000G	00	03	FB 00BD2	CALLS	#3, LIB\$SIGNAL	
				37	11 00BD9	BRB	157\$
			50	AD	9E 00BDB	152\$:	MOVAB INTMED_DATA, R0
			51	AE	DO 00BDF	MOVL	OUTPUT, R1
			00000000G	00	16 00BE3	JSB	DBG\$CVT_CVTHG_R1
				08	C1 00BE9	ADDL3	#8, OUTPUT, RT
51	34	AE		AD	9E 00BEE	MOVAB	INTMED_DATA+16, R0
			C0	10	11 00BF2	BRB	155\$
34	BE	B0	AD	20	28 00BF4	153\$:	MOV C3 #32, INTMED_DATA, @OUTPUT
				16	11 00BFA	BRB	157\$
			50	AD	9E 00BFC	154\$:	MOVAB INTMED_DATA, R0
			51	AE	DO 00C00	MOVL	OUTPUT, R1
			00000000G	00	16 00C04	155\$:	JSB DBG\$CVT_CVTHG_R1
				06	11 00C0A	BRB	157\$
34	BE	B0	AD	10	28 00C0C	156\$:	MOV C3 #16, INTMED_DATA, @OUTPUT

2463

2473

2488

2489

2490

2491

2492

2493

2494

2495

2496

2497

2486

2478

2479

2483

2473

2467

2470

			06	FF	AD	E9	00C12	157\$:	BLBC	SRC_INFO+7, 158\$	2502
		34	BE	8000	8F	A8	00C16		BISW2	#32768, @OUTPUT	
					23B9	31	00C1C	158\$:	BRW	649\$	2187
			05		56	D1	00C1F	159\$:	CMPL	R6, #5	2510
					03	13	00C22		BEQL	160\$	
					00C2	31	00C24		BRW	168\$	
				B0	AD	D5	00C27	160\$:	TSTL	INTMED_DATA	2511
					04	18	00C2A		BGEQ	161\$	
		FF	AD		01	88	00C2C		BISB2	#1, SRC_INFO+7	
		2C	AE		1F	D0	00C30	161\$:	MOVL	#31, NO_DIGITS	
B0	AD	2C	AE	B0	AD	F9	00C34		CVTLP	INTMED_DATA, NO_DIGITS, INTMED_DATA	
				30	AE	D5	00C3B		TSTL	SCALE	
					3C	13	00C3E		BEQL	164\$	
FF7C	CD	B0	AD	2C	AE	34	00C40		MOVP	NO_DIGITS, INTMED_DATA, TEMP_BUF1	
			OE	0C	AE	E9	00C48		BLBC	CVT_ROUND_FLAG, 162\$	
			55	B0	AD	9E	00C4C		MOVAB	INTMED_DATA, R5	
			54	2C	AE	9E	00C50		MOVAB	NO_DIGITS, R4	
		08	AE		05	D0	00C54		MOVL	#5, 8(SP)	
					0B	11	00C58		BRB	163\$	
			55	B0	AD	9E	00C5A	162\$:	MOVAB	INTMED_DATA, R5	
			54	2C	AE	9E	00C5E		MOVAB	NO_DIGITS, R4	
				08	AE	D4	00C62		CLRL	8(SP)	
			53	08	AE	9E	00C65	163\$:	MOVAB	8(SP), R3	
			52	FF7C	CD	9E	00C69		MOVAB	TEMP_BUF1, R2	
			51	2C	AE	9E	00C6E		MOVAB	NO_DIGITS, R1	
			50	30	AE	9E	00C72		MOVAB	SCALE, R0	
				00000000G	00	16	00C76		JSB	DBG\$CVT_ASH_P_R1	
					58	D5	00C7C	164\$:	TSTL	BIN_SCALE	
					32	15	00C7E		BLEQ	165\$	
FF7C	CD	B0	AD	2C	AE	34	00C80		MOVP	NO_DIGITS, INTMED_DATA, TEMP_BUF1	
			55	B0	AD	9E	00C88		MOVAB	INTMED_DATA, R5	
			54	2C	AE	9E	00C8C		MOVAB	NO_DIGITS, R4	
			53	FF7C	CD	9E	00C90		MOVAB	TEMP_BUF1, R3	
			52	2C	AE	9E	00C95		MOVAB	NO_DIGITS, R2	
			51	00000000'	EF	9E	00C99		MOVAB	P_AGP, R1	
		08	AE		01	D0	00CA0		MOVL	#1, 8(SP)	
			50	08	AE	9E	00CA4		MOVAB	8(SP), R0	
				00000000G	00	16	00CA8		JSB	DBG\$CVT_MULP_R1	
					58	D7	00CAE		DECL	BIN_SCALE	
					CA	11	00CB0		BRB	164\$	
					03	19	00CB2	165\$:	BLSS	167\$	
					0105	31	00CB4	166\$:	BRW	175\$	
FF7C	CD	B0	AD	2C	AE	34	00CB7	167\$:	MOVP	NO_DIGITS, INTMED_DATA, TEMP_BUF1	
			55	B0	AD	9E	00CBF		MOVAB	INTMED_DATA, R5	
			54	2C	AE	9E	00CC3		MOVAB	NO_DIGITS, R4	
			53	FF7C	CD	9E	00CC7		MOVAB	TEMP_BUF1, R3	
			52	2C	AE	9E	00CCB		MOVAB	NO_DIGITS, R2	
			51	00000000'	EF	9E	00CD0		MOVAB	P_AGP, R1	
		08	AE		01	D0	00CD7		MOVL	#1, 8(SP)	
			50	08	AE	9E	00CDB		MOVAB	8(SP), R0	
				00000000G	00	16	00CDF		JSB	DBG\$CVT_DIVP_R1	
					58	D6	00CE5		INCL	BIN_SCALE	
					C9	11	00CE7		BRB	165\$	
			1D		56	D1	00CE9	168\$:	CMPL	R6, #29	2515
					C6	12	00CEC		BNEQ	166\$	
08	BE		2C	AE	FD	AD	3C	00CEE	MOVZWL	SRC_INFO+5, NO_DIGITS	2516
		01	B0	AD	2C	AE	37	00CF3	CMPP4	NO_DIGITS, INTMED_DATA, #1, @PACK_ZERO	



2521

00ED	00CD	00CD	00DC8				
				181\$-176\$,-			
				184\$-176\$,-			
				188\$-176\$,-			
				192\$-176\$,-			
				192\$-176\$,-			
				196\$-176\$,-			
	05	56	D1 00DCE	CMPL	R6, #5		2574
		08	12 00DD1	BNEQ	177\$		
	00000000'	EF	9F 00DD3	PUSHAB	P.AGQ		2575
		0B	11 00DD9	BRB	178\$		
	1D	56	D1 00DD8 177\$:	CMPL	R6, #29		2576
		7D	12 00DDE	BNEQ	187\$		
	00000000'	EF	9F 00DE0	PUSHAB	P.AGR		2577
		21E0	31 00DE6 178\$:	BRW	647\$		
	15	FF	AD E9 00DE9 179\$:	BLBC	SRC INFO+7, 180\$		2526
	00000000G	00	DD 00DED	PUSHL	DBG\$GL_OPCODE_NAME		
		01	DD 00DF3	PUSHL	#1		
	00028EF0	8F	DD 00DF5	PUSHL	#167664		
		03	FB 00DFB	CALLS	#3, LIB\$SIGNAL		
69 00000000G	00	00000000G	00	CVTPT	NO DIGITS, INTMED_DATA, LIB\$AB_CVTPT_U, -		2527
		2C	AE 24 00E02 180\$:		(R9), @OUTPUT		
		34	BE				
		7A	11 00E0F	BRB	191\$		2521
		69	B5 00E11 181\$:	TSTW	(R9)		2533
		04	12 00E13	BNEQ	182\$		
		50	D4 00E15	CLRL	R0		
		05	11 00E17	BRB	183\$		
	50	69	3C 00E19 182\$:	MOVZWL	(R9), R0		
		50	67 00E1C	DECL	R0		
34	BE	50	B0 AD 2C	AE 08 00E1E 183\$:	CVTPS	NO DIGITS, INTMED_DATA, R0, @OUTPUT	2534
				0082	31 00E26	BRW	195\$
69 00000000G	00	B0	AD 2C	AE 24 00E29 184\$:	CVTPT	NO DIGITS, INTMED_DATA, LIB\$AB_CVTPT_U, -	2538
						(R9), TEMP_BUF1	
		50	FF7C	CD	9A 00E37	MOVZBL	TEMP_BUF1, R0
		50	FF7C	CD	9E 00E3C	MOVAB	LIB\$AB_CVTPT_U 0-48[R0], R0
		06	FF	AD	E9 00E44	BLBC	SRC INFO+7, T85\$
		50	0A	AD	D0 00E48	MOVL	10(R0), R0
				03	11 00E4C	BRB	186\$
		50	60	D0 00E4E 185\$:	MOVL	(R0), R0	2540
			50	90 00E51 186\$:	MOVB	R0, TEMP_BUF1	2539
34	BE	FF7C	CD	69	28 00E56	MOVBC3	(R9), TEMP_BUF1, @OUTPUT
		FF7C	CD	61	11 00E5D 187\$:	BRB	197\$
				69	B5 00E5F 188\$:	TSTW	(R9)
				04	12 00E61	BNEQ	189\$
				57	D4 00E63	CLRL	DES_LEN
				05	11 00E65	BRB	190\$
	57	69	3C 00E67 189\$:	MOVZWL	(R9), DES_LEN		
		57	D7 00E6A	DECL	DES_LEN		
FF7C	CD	57	B0 AD 2C	AE 08 00E6C 190\$:	CVTPS	NO DIGITS, INTMED_DATA, DES_LEN, TEMP_BUF1	2552
					90 00E75	MOVB	TEMP_BUF1, INTMED_DATA[DES_LEN]
	B0	AD	FF7D	CD	57	28 00E7C	MOVBC3
					57	D6 00E83	INCL
					57	28 00E85	MOVBC3
34	BE	B0	AD	33	11 00E8B 191\$:	BRB	197\$
				13	6B 91 00E8D 192\$:	CMPB	(R11), #19
					09	12 00E90	BNEQ
					50	00000000G	00
					07	11 00E99	MOVAB
							LIB\$AB_CVTPT_0, R0
							BRB
							194\$

69	60	B0	AD	50	00000000G	00	9E	00E9B	193\$:	MOVAB	LIB\$AB_CVTPT_2, R0	
					2C	AE	24	00EA2	194\$:	CVTPT	NO_DIGITS, INTMED_DATA, (R0), (R9), @OUTPUT	2561
					34	BE		00EA9				
						13	11	00EAB	195\$:	BRB	197\$	
FF7C	CD	2C	AE	B0	AD	2C	AE	08	00EAD	196\$:	CVTPS	NO DIGITS, INTMED_DATA, NO_DIGITS, -
											TEMP_BUF1	2567
34	BE	69	FF7C	CD	2C	AE	09	00EB7		CVTSP	NO DIGITS, TEMP_BUF1, (R9), @OUTPUT	2568
					2115	31	00EC0	197\$:	BRW	649\$		2187
					30	AE	D5	00EC3	198\$:	TSTL	SCALE	2583
						0F	15	00EC6		BLEQ	199\$	
				50	B0	AD	9E	00EC8		MOVAB	INTMED_DATA, R0	
					00000000G	00	16	00ECC		JSB	LIB\$SCVT_SCALE_OU_UP_BY_10_R1	
					30	AE	D7	00ED2		DECL	SCALE	
						EC	11	00ED5		BRB	198\$	
						0F	18	00ED7	199\$:	BGEQ	200\$	
				50	B0	AD	9E	00ED9		MOVAB	INTMED_DATA, R0	
					00000000G	00	16	00EDD		JSB	LIB\$SCVT_SCALE_OU_DOWN_BY_10_R1	
					30	AE	D6	00EE3		INCL	SCALE	
						EF	11	00EE6		BRB	199\$	
						58	D5	00EE8	200\$:	TSTL	BIN SCALE	
						0E	15	00EEA		BLEQ	201\$	
				50	B0	AD	9E	00EEC		MOVAB	INTMED_DATA, R0	
					00000000G	00	16	00EF0		JSB	DBG\$CVT_SCALE_OU_UP_BY_2_R1	
						58	D7	00EF6		DECL	BIN SCALE	
						EE	11	00EF8		BRB	200\$	
						0E	18	00EFA	201\$:	BGEQ	202\$	
				50	B0	AD	9E	00EFC		MOVAB	INTMED_DATA, R0	
					00000000G	00	16	00F00		JSB	DBG\$CVT_SCALE_OU_DOWN_BY_2_R1	
						58	D6	00F06		INCL	BIN SCALE	
						F0	11	00F08		BRB	201\$	
				50	B4	AD	C9	00F0A	202\$:	BISL3	INTMED_DATA+8, INTMED_DATA+4, R0	2585
					50	BC	C8	00F10		BISL2	INTMED_DATA+12, R0	
						15	13	00F14		BEQL	203\$	
					00000000G	00	DD	00F16		PUSHL	DBG\$GL_OPCODE_NAME	2587
						01	DD	00F1C		PUSHL	#1	
					000286A3	8F	DD	00F1E		PUSHL	#165539	
						03	FB	00F24		CALLS	#3, LIB\$SIGNAL	
						6B	8F	00F2B	203\$:	CASEB	(R11), #1, #41	2588
								00F2F	204\$:	.WORD		
0054	0082	005D	0130	0054	00F37	0054	00F3F	0054	00F47	205\$-204\$,-		
00F8	00CE	00A4	0054	0054	00F4F	0054	00F57	0054	00F5F	206\$-204\$,-		
0054	0054	0054	0054	0054	00F67	0054	00F6F	0054	00F77	207\$-204\$,-		
0054	0054	0054	0054	0054	00F7F	0054	00F8F	0054	00F9F	208\$-204\$,-		
0054	0054	0054	0054	0054	00FAF	0054	00FBF	0054	00FCF	209\$-204\$,-		
0054	0054	0054	0054	0054	00FDF	0054	00FEF	0054	00FFF	210\$-204\$,-		
0054	0054	0054	0054	0054	00F0F	0054	00F1F	0054	00F2F	211\$-204\$,-		
0054	0054	0054	0054	0054	00F3F	0054	00F4F	0054	00F5F	212\$-204\$,-		
0054	0054	0054	0054	0054	00F6F	0054	00F7F	0054	00F8F	213\$-204\$,-		
0054	0054	0054	0054	0054	00F9F	0054	00FAF	0054	00FBF	214\$-204\$,-		
0054	0054	0054	0054	0054	00FCF	0054	00FDF	0054	00FEF	215\$-204\$,-		
0054	0054	0054	0054	0054	00FFF	0054	0100F	0054	0101F	216\$-204\$,-		
0054	0054	0054	0054	0054	0102F	0054	0103F	0054	0104F	217\$-204\$,-		
0054	0054	0054	0054	0054	0105F	0054	0106F	0054	0107F	218\$-204\$,-		
0054	0054	0054	0054	0054	0108F	0054	0109F	0054	010AF	219\$-204\$,-		
0054	0054	0054	0054	0054	010BF	0054	010CF	0054	010DF	220\$-204\$,-		
0054	0054	0054	0054	0054	010EF	0054	010FF	0054	0110F	221\$-204\$,-		
0054	0054	0054	0054	0054	0111F	0054	0112F	0054	0113F	222\$-204\$,-		
0054	0054	0054	0054	0054	0114F	0054	0115F	0054	0116F	223\$-204\$,-		
0054	0054	0054	0054	0054	0117F	0054	0118F	0054	0119F	224\$-204\$,-		
0054	0054	0054	0054	0054	011AF	0054	011BF	0054	011CF	225\$-204\$,-		
0054	0054	0054	0054	0054	011DF	0054	011EF	0054	011FF	226\$-204\$,-		
0054	0054	0054	0054	0054	0120F	0054	0121F	0054	0122F	227\$-204\$,-		
0054	0054	0054	0054	0054	0123F	0054	0124F	0054	0125F	228\$-204\$,-		
0054	0054	0054	0054	0054	0126F	0054	0127F	0054	0128F	229\$-204\$,-		
0054	0054	0054	0054	0054	0129F	0054	012AF	0054	012BF	230\$-204\$,-		
0054	0054	0054	0054	0054	012CF	0054	012DF	0054	012EF	231\$-204\$,-		
0054	0054	0054	0054	0054	012FF	0054	0130F	0054	0131F	232\$-204\$,-		
0054	0054	0054	0054	0054	0132F	0054	0133F	0054	0134F	233\$-204\$,-		
0054	0054	0054	0054	0054	0135F	0054	0136F	0054	0137F	234\$-204\$,-		
0054	0054	0054	0054	0054	0138F	0054	0139F	0054	013AF	235\$-204\$,-		
0054	0054	0054	0054	0054	013BF	0054	013CF	0054	013DF	236\$-204\$,-		
0054	0054	0054	0054	0054	013EF	0054	013FF	0054	0140F	237\$-204\$,-		
0054	0054	0054	0054	0054	0141F	0054	0142F	0054	0143F	238\$-204\$,-		
0054	0054	0054	0054	0054	0144F	0054	0145F	0054	0146F	239\$-204\$,-		
0054	0054	0054	0054	0054	0147F	0054	0148F	0054	0149F	240\$-204\$,-		
0054	0054	0054	0054	0054	014AF	0054	014BF	0054	014CF	241\$-204\$,-		
0054	0054	0054	0054	0054	014DF	0054	014EF	0054	014FF	242\$-204\$,-		
0054	0054	0054	0054	0054	0150F	0054	0151F	0054	0152F	243\$-204\$,-		
0054	0054	0054	0054	0054	0153F	0054	0154F	0054	0155F	244\$-204\$,-		
0054	0054	0054	0054	0054	0156F	0054	0157F	0054	0158F	245\$-204\$,-		
0054	0054	0054	0054	0054	0159F	0054	015AF	0054	015BF	246\$-204\$,-		
0054	0054	0054	0054	0054	015CF	0054	015DF	0054	015EF	247\$-204\$,-		
0054	0054	0054	0054	0054	015FF	0054	0160F	0054	0161F	248\$-204\$,-		
0054	0054	0054	0054	0054	0162F	0054	0163F	0054	0164F	249\$-204\$,-		
0054	0054	0054	0054	0054	0165F	0054	0166F	0054	0167F	250\$-204\$,-		
0054	0054	0054	0054	0054	0168F	0054	0169F	0054	016AF	251\$-204\$,-		
0054	0054	0054	0054	0054	016BF	0054	016CF	0054	016DF	252\$-204\$,-		
0054	0054	0054	0054	0054	016EF	0054	016FF	0054	0170F	253\$-204\$,-		
0054	0054	0054	0054	0054	0171F	0054	0172F	0054	0173F	254\$-204\$,-		
0054	0054	0054	0054	0054	0174F	0054	0175F	0054	0176F	255\$-204\$,-		
0054	0054	0054	0054	0054	0177F	0054	0178F	0054	0179F	256\$-204\$,-		
0054	0054	0054	0054	0054	017AF	0054	017BF	0054	017CF	257\$-204\$,-		
0054	0054	0054	0054	0054	017DF	0054	017EF	0054	017FF	258\$-204\$,-		
0054	0054	0054	0054	0054	0180F	0054	0181F	0054	0182F	259\$-204\$,-		
0054	0054	0054	0054	0054	0183F	0054	0184F	0054	0185F	260\$-204\$,-		
0054	0054	0054	0054	0054	0186F	0054	0187F	0054	0188F	261\$-204\$,-		
0054	0054	0054	0054	0054	0189F	0054	018AF	0054	018BF	262\$-204\$,-		
0054	0054	0054	0054	0054	018CF	0054	018DF	0054	018EF	263\$-204\$,-		
0054	0054	0054	0054	0054	018FF	0054	0190F	0054	0191F	264\$-204\$,-		
0054	0054	0054	0054	0054	0192F	0054	0193F	0054	0194F	265\$-204\$,-		
0054	0054	0054	0054	0054	0195F	0054	0196F	0054	0197F	266\$-204\$,-		
0054	0054	0054	0054	0054	0198F	0054	0199F	0054	019AF	267\$-204\$,-		
0054	0054	0054	0054	0054	019BF	0054	019CF	0054	019DF	268\$-204\$,-		
0054	0054	0054	0054	0054	019EF	0054	019FF	0054	01A0F	269\$-204\$,-		
0054	0054	0054	0054	0054	01A1F	0054	01A2F	0054	01A3F	270\$-204\$,-		
0054	0054	0054	0054	0054	01A4F	0054	01A5F	0054	01A6F	271\$-204\$,-		
0054	0054	0054	0054	0054	01A7F	0054	01A8F	0054	01A9F	272\$-204\$,-		
0054	0054	0054	0054	0054	01AAF	0054	01ABF	0054	01ACF	273\$-204\$,-		
0054	0054	0054	0054	0054	01ADF	0054	01AEF	0054	01AFF	274\$-204\$,-		
0054	0054	0054	0054	0054	01B0F	0054	01B1F	0054	01B2F	275\$-204\$,-		
0054	0054	0054	0054	0054	01B3F	0054	01B4F	0054	01B5F	276\$-204\$,-		
0054	0054	0054	0054	0054	01B6F	0054	01B7F	0054	01B8F	277\$-204\$,-		
0054	0054	0054	0054	0054	01B9F	0054	01BAF	0054	01BBF	278\$-204\$,-		
0054	0054	0054	0054	0054	01BBF	0054	01BCF	0054	01BCF	279\$-204\$,-		
0054	0054	0054	0									

Address	Disassembly	Comment	Symbol
00000000	EF 9F 00F83	205\$: PUSHAB	P.AGS
05	203D 31 00F89	BRW	647\$
B1	AD E8 00F8C	206\$: BLBS	INTMED_DATA+1, 207\$
B2	AD B5 00F90	TSTW	INTMED_DATA+2
	15 13 00F93	BEQL	208\$
00000000G	00 DD 00F95	207\$: PUSHL	DBG\$GL_OPCODE_NAME
	01 DD 00F98	PUSHL	#1
000286A3	8F DD 00F9D	PUSHL	#165539
00	03 FB 00FA3	CALLS	#3, LIB\$SIGNAL
34	BE B0 AD 90 00FAA	208\$: MOVW	INTMED_DATA, @OUTPUT
	1F 11 00FAF	BRB	211\$
	B2 AD B5 00FB1	209\$: TSTW	INTMED_DATA+2
	15 13 00FB4	BEQL	210\$
00000000G	00 DD 00FB6	PUSHL	DBG\$GL_OPCODE_NAME
	01 DD 00FBC	PUSHL	#1
000286A3	8F DD 00FBE	PUSHL	#165539
00	03 FB 00FC4	CALLS	#3, LIB\$SIGNAL
34	BE B0 AD B0 00FCB	210\$: MOVW	INTMED_DATA, @OUTPUT
	008A 31 00FD0	211\$: BRW	222\$
	B0 AD D5 00FD3	212\$: TSTL	INTMED_DATA
	15 18 00FD6	BGEQ	213\$
00000000G	00 DD 00FD8	PUSHL	DBG\$GL_OPCODE_NAME
	01 DD 00FDE	PUSHL	#1
000286A3	8F DD 00FE0	PUSHL	#165539
00	03 FB 00FE6	CALLS	#3, LIB\$SIGNAL
05	FF AD E9 00FED	213\$: BLBC	SRC INFO+7, 214\$
B0	AD CE 00FF1	MNEGL	INTMED_DATA, INTMED_DATA
50	B0 AD 9E 00FF6	214\$: MOVAB	INTMED_DATA, R0
	18FB 31 00FFA	BRW	540\$
	B0 AD D5 00FFD	215\$: TSTL	INTMED_DATA
	15 18 01000	BGEQ	216\$
00000000G	00 DD 01002	PUSHL	DBG\$GL_OPCODE_NAME
	01 DD 01008	PUSHL	#1
000286A3	8F DD 0100A	PUSHL	#165539

00000000G	00	03	FB	01010	CALLS	#3, LIB\$SIGNAL	
	05	FF	AD	E9 01017	216\$:	BLBC	SRC_INFO+7, 217\$
B0	AD	B0	AD	CE 0101B		MNEGL	INTMED_DATA, INTMED_DATA
	50	B0	AD	9E 01020	217\$:	MOVAB	INTMED_DATA, R0
80000000	8F	B0	AD	31 01024		BRW	551\$
				D1 01027	218\$:	CMPL	INTMED_DATA, #-2147483648
	23	FF	AD	12 0102F		BNEQ	219\$
		B0	AD	E8 01031		BLBS	SRC_INFO+7, 221\$
				D5 01035	219\$:	TSTL	INTMED_DATA
		15	18	01038		BGEQ	220\$
00000000G	00	00	DD	0103A		PUSHL	DBG\$GL_OPCODE_NAME
	01	DD	DD	01040		PUSHL	#1
000286A3	8F	DD	DD	01042		PUSHL	#165539
00000000G	00	03	FB	01048		CALLS	#3, LIB\$SIGNAL
	05	FF	AD	E9 0104F	220\$:	BLBC	SRC_INFO+7, 221\$
B0	AD	B0	AD	CE 01053		MNEGL	INTMED_DATA, INTMED_DATA
34	BE	B0	AD	D0 01058	221\$:	MOVL	INTMED_DATA, @OUTPUT
		19	11	0105D	222\$:	BRB	226\$
	52	F5	AD	3C 0105F	223\$:	MOVZWL	DST_INFO+5, R2
	50		01	CE 01063		MNEGL	#1, I
			0C	11 01066		BRB	225\$
34	51	B0	AD	50 EF 01068	224\$:	EXTZV	I, #1, INTMED_DATA, R1
BE			51	FO 0106E		INSV	R1, I, #1, @OUTPUT
			52	F2 01074	225\$:	A0BLSS	R2, I, 224\$
			5D	31 01078	226\$:	BRW	646\$
58	AE		32	B0 0107B	227\$:	MOVW	#50, CLASS_S_DESC
5C	AE	60	AE	9E 0107F		MOVAB	TEMP_BUF2, -CLASS_S_DESC+4
	0B		56	D1 01084		CMPL	R6, #11
			23	12 01087		BNEQ	229\$
	51	FF7C	CD	9E 01089		MOVAB	TEMP_BUF1, R1
	50	B0	AD	9E 0108E		MOVAB	INTMED_DATA, R0
		00000000G	00	16 01092		JSB	DBG\$CVT_CVTROUH R1
	06	FF	AD	E9 01098		BLBC	SRC_INFO+7, 228\$
FF7D	CD	B0	8F	88 0109C		BISB2	#128, TEMP_BUF1+1
			01	DD 010A2	228\$:	PUSHL	#1
			7E	7C 010A4		CLRQ	-(SP)
		3C	AE	DD 010A6		PUSHL	SCALE
			31	010A9		BRW	239\$
	11		56	D1 010AC	229\$:	CMPL	R6, #17
			21	12 010AF		BNEQ	231\$
		B1	AD	95 010B1		TSTB	INTMED_DATA+1
			04	18 010B4		BGEQ	230\$
	FF	AD	01	88 010B6		BISB2	#1, SRC_INFO+7
			01	DD 010BA	230\$:	PUSHL	#1
			7E	7C 010BC		CLRQ	-(SP)
		3C	AE	DD 010BE		PUSHL	SCALE
			7E	D4 010C1		CLRL	-(SP)
		6C	AE	9F 010C3		PUSHAB	CLASS_S_DESC
		B0	AD	9F 010C6		PUSHAB	INTMED_DATA
00000000G	00		07	FB 010C9		CALLS	#7, FOR\$CVT_D_TF
			30	11 010D0		BRB	234\$
	17		56	D1 010D2	231\$:	CMPL	R6, #23
			3E	12 010D5		BNEQ	236\$
		B1	AD	95 010D7		TSTB	INTMED_DATA+1
			04	18 010DA		BGEQ	232\$
	FF	AD	01	88 010DC		BISB2	#1, SRC_INFO+7
	1B	04	BE	91 010E0	232\$:	CMPB	@4(SP), -#27

			06	13	010E4	BEQL	233\$		
	1D	04	BE	91	010E6	CMPB	24(SP), #29		2672
			18	12	010EA	BNEQ	235\$		
			01	DD	010EC	233\$: PUSHL	#1		2674
			7E	7C	010EE	CLRQ	-(SP)		
		3C	AE	DD	010F0	PUSHL	SCALE		
			7E	D4	010F3	CLRL	-(SP)		
		6C	AE	9F	010F5	PUSHAB	CLASS_S_DESC		
		80	AD	9F	010F8	PUSHAB	INTMED_DATA		
00000000G	00		07	FB	010FB	CALLS	#7, FOR\$CVT_G_TF		
			7C	11	01102	234\$: BRB	241\$		
			01	DD	01104	235\$: PUSHL	#1		2676
			7E	7C	01106	CLRQ	-(SP)		
		3C	AE	DD	01108	PUSHL	SCALE		
			7E	D4	0110B	CLRL	-(SP)		
		6C	AE	9F	0110D	PUSHAB	CLASS_S_DESC		
		80	AD	9F	01110	PUSHAB	INTMED_DATA		
			64	11	01113	BRB	240\$		
	23		56	D1	01115	236\$: CMPL	R6, #35		2679
			69	12	01118	BNEQ	242\$		
58	AE	FD	AD	B0	0111A	MOVW	SRC_INFO+5, CLASS_S_DESC		2681
5C	AE	F9	AD	D0	0111F	MOVL	SRC_INFO+1, CLASS_S_DESC+4		2682
	7E	55	8F	9A	01124	MOVZBL	#85, -(SP)		2684
	7E	34	AE	CE	01128	MNEGL	SCALE, -(SP)		2683
			7E	D4	0112C	CLRL	-(SP)		
		FF7C	CD	9F	0112E	PUSHAB	TEMP_BUF1		
		68	AE	9F	01132	PUSHAB	CLASS_S_DESC		
00000000G	00		05	FB	01135	CALLS	#5, OTS\$CVT_T_H		
	6E		50	D0	0113C	MOVL	R0, STATUS		
	15		6E	E8	0113F	BLBS	STATUS, 237\$		2685
		00000000G	00	DD	01142	PUSHL	DBG\$GL_OPCODE_NAME		
			01	DD	01148	PUSHL	#1		
		00028298	8F	DD	0114A	PUSHL	#164504		
00000000G	00		03	FB	01150	CALLS	#3, LIB\$SIGNAL		
		FF7D	CD	95	01157	237\$: TSTB	TEMP_BUF1+1		2686
			04	18	0115B	BGEQ	238\$		
	FF	AD	01	88	0115D	BISB2	#1, SRC_INFO+7		
58	AE		32	B0	01161	238\$: MOVW	#50, CLASS_S_DESC		2687
5C	AE	60	AE	9E	01165	MOVAB	TEMP_BUF2, CLASS_S_DESC+4		2688
			01	DD	0116A	PUSHL	#1		2689
			7E	7C	0116C	CLRQ	-(SP)		
			7E	D4	0116E	CLRL	-(SP)		
			7E	D4	01170	239\$: CLRL	-(SP)		
		6C	AE	9F	01172	PUSHAB	CLASS_S_DESC		
		FF7C	CD	9F	01175	PUSHAB	TEMP_BUF1		
00000000G	00		07	FB	01179	240\$: CALLS	#7, FOR\$CVT_H_TF		
	6E		50	D0	01180	241\$: MOVL	R0, STATUS		
	15		6E	E8	01183	242\$: BLBS	STATUS, 243\$		2693
		00000000G	00	DD	01186	PUSHL	DBG\$GL_OPCODE_NAME		
			01	DD	0118C	PUSHL	#1		
		00028A3A	8F	DD	0118E	PUSHL	#166458		
00000000G	00		03	FB	01194	CALLS	#3, LIB\$SIGNAL		
60	AE		20	3B	0119B	243\$: SKPC	#32, #50, TEMP_BUF2		2694
			02	12	011A0	BNEQ	244\$		
			51	D4	011A2	CLRL	R1		
		50	AE	9E	011A4	244\$: MOVAB	TEMP_BUF2, R0		
5A		51	50	C3	011A8	SUBL3	R0, R1, BUF_OFFSET		

0130	2C AE 06 00DA 01D3	30 OF 009E 018C	SA 6B 0046 018C	C3 011AC 8F 011B1 011B5 011B0	SUBL3 CASEB .WORD	BUF_OFFSET, #48, NO_DIGITS (R1T), #15, #6 252\$-245\$,- 255\$-245\$,- 259\$-245\$,- 264\$-245\$,- 270\$-245\$,- 270\$-245\$,- 274\$-245\$	2695 2697
		08	56	D1 011C3	CMPL	R6, #11	2768
			08	12 011C6	BNFQ	246\$	
		00000000'	EF	9F 011C8	PUSHAB	P.AGT	2769
			29	11 011CE	BRB	251\$	
		11	56	D1 011D0	CMPL	R6, #17	2770
			08	12 011D3	BNEQ	247\$	
		00000000'	EF	9F 011D5	PUSHAB	P.AGU	2771
			1C	11 011DB	BRB	251\$	
		17	56	D1 011DD	CMPL	R6, #23	2772
			09	12 011E0	BNEQ	249\$	
		00000000'	EF	9F 011E2	PUSHAB	P.AGV	2773
			1DDE	31 011E8	BRW	647\$	
		23	56	D1 011EB	CMPL	R6, #35	2774
			03	13 011EE	BEQL	250\$	
		00000000'	1DES	31 011F0	BRW	649\$	
			EF	9F 011F3	PUSHAB	P.AGW	2775
			ED	11 011F9	BRB	248\$	
		15	FF	E9 011FB	BLBC	SRC_INFO+7, 253\$	2702
		00000000G	00	DD 011FF	PUSHL	DBG\$GL_OPCODE_NAME	
			01	DD 01205	PUSHL	#1	
		00028EF0	8F	DD 01207	PUSHL	#167664	
			03	FB 0120D	CALLS	#3, LIB\$SIGNAL	
		57	69	3C 01214	MOVZWL	(R9), R7	2703
		57	2C	AE D1 01217	CMPL	NO DIGITS, R7	
			15	15 0121B	BLEQ	254\$	
		00000000G	00	DD 0121D	PUSHL	DBG\$GL_OPCODE_NAME	
			01	DD 01223	PUSHL	#1	
		00028A3A	8F	DD 01225	PUSHL	#166458	
			03	FB 0122B	CALLS	#3, LIB\$SIGNAL	
		50	2C	AE C3 01232	SUBL3	NO DIGITS, R7, R0	2704
		30	6E	00 2C 01237	MOVCS	#0, (SP), #48, R0, TEMP_BUF1	
			50	CD 0123C	MOVAB	TEMP_BUF1[R7], R0	2706
			50	2C AE C2 01245	SUBL2	NO DIGITS, R0	
		60	61 AE4A	2C AE 28 01249	MOVCS	NO DIGITS, TEMP_BUF2+1[BUF_OFFSET], (R0)	
			00E5	31 01250	BRW	268\$	2707
			69	B5 01253	TSTW	(R9)	2716
			04	12 01255	BNEQ	256\$	
			54	D4 01257	CLRL	DES_LEN	
			05	11 01259	BRB	257\$	
		54	69	3C 0125B	MOVZWL	(R9), DES_LEN	
			54	D7 0125E	DECL	DES_LEN	
		2C AE	54	D1 01260	CMPI	DES_LEN, NO_DIGITS	2718
			15	18 01264	BGR	258\$	
		00000000G	00	DD 01266	PL HL	DBG\$GL_OPCODE_NAME	
			01	DD 0126C	PUSHL	#1	
		00028A3A	8F	DD 0126E	PUSHL	#166458	
			03	FB 01274	CALLS	#3, LIB\$SIGNAL	

FF7C	CD	54	60	AE4A	2C	AE	09	0127B	258\$:	CVTSP	NO DIGITS, TEMP_BUF2[BUF_OFFSET], DES_LEN, -;	2719	
											TEMP_BUF1		
34	BE	54	FF7C	CD		54	08	01285		CVTPS	DES_LEN, TEMP_BUF1, DES_LEN, @OUTPUT	2720	
				52	01	54	11	0128D		BRB	263\$	2697	
				6E		AA	9E	0128F	259\$:	MOVAB	1(R10), R2	2725	
	52	30				00	2C	01293		MOVCS	#0, (SP), #48, R2, TEMP_BUF2		
					60	AE		01298					
2C	AE	69		10		00	ED	0129A		CMPZV	#0, #16, (R9), NO_DIGITS	2726	
						15	18	012A0		BGEQ	260\$		
			00000000G			00	DD	012A2		PUSHL	DBG\$GL_OPCODE_NAME		
						01	DD	012A8		PUSHL	#1		
			00028A3A			8F	DD	012AA		PUSHL	#166458		
		00000000G	00			03	FB	012B0		CALLS	#3, LIB\$SIGNAL		
5A			5A			69	3C	012B7	260\$:	MOVZWL	(R9), BUF_OFFSET	2727	
			31			5A	C3	012BA		SUBL3	BUF_OFFSET, #49, BUF_OFFSET		
			51		60	AE4A	9E	012BE		MOVAB	TEMP_BUF2[BUF_OFFSET], R1	2728	
			50			61	9A	012C3		MOVZBL	(R1), R0		
			50	00000000G	00	40	9E	012C6		MOVAB	LIB\$AB CVT_U 0-48(R0), R0	2729	
			06		FF	AD	E9	012CE		BLBC	SRC INFO+7, 261\$	2728	
			50		0A	A0	D0	012D2		MOVL	10(R0), R0		
						03	11	012D6		BRB	262\$		
			50			60	D0	012D8	261\$:	MOVL	(R0), R0	2729	
			61			50	90	012DB	262\$:	MOVB	R0, (R1)	2728	
34	BE		61			69	28	012DE		MOVCS	(R9), (R1), @OUTPUT	2731	
						5A	11	012E3	263\$:	BRB	269\$	2697	
						69	B5	012E5	264\$:	TSTW	(R9)	2740	
						04	12	012E7		BNEQ	265\$		
						57	D4	012E9		CLRL	DES_LEN		
						05	11	012EB		BRB	266\$		
			57			69	3C	012ED	265\$:	MOVZWL	(R9), DES_LEN		
						57	D7	012F0		DECL	DES_LEN		
			57		2C	AE	D1	012F2	266\$:	CMP	NO DIGITS, DES_LEN	2742	
						15	15	012F6		BLEQ	267\$		
				00000000G		00	DD	012F8		PUSHL	DBG\$GL_OPCODE_NAME		
						01	DD	012FE		PUSHL	#1		
				00028A3A		8F	DD	01300		PUSHL	#166458		
		00000000G	00			03	FB	01306		CALLS	#3, LIB\$SIGNAL		
50			57		2C	AE	C3	0130D	267\$:	SUBL3	NO_DIGITS, DES_LEN, R0	2743	
						50	D6	01312		INCL	R0		
	50		30			6E	00	2C	01314	MOVCS	#0, (SP), #48, R0, TEMP_BUF1		
					FF7C	CD			01319				
				10	AE	FF7C	CD	47	9E	0131C	MOVAB	TEMP_BUF1[DES_LEN], 16(SP)	2744
			50			2C	AE	C3	01323	SUBL3	NO_DIGITS, 16(SP), R0		
			60		61	AE4A	2C	AE	28	01329	MOVCS	NO_DIGITS, TEMP_BUF2+1[BUF_OFFSET], (R0)	
				10	BE	60	AE4A	90	01330	MOVB	TEMP_BUF2[BUF_OFFSET], @16(SP)	2745	
						57	D6	01336		INCL	R7	2747	
34	BE		FF7C	CD		57	28	01338	268\$:	MOVCS	R7, TEMP_BUF1, @OUTPUT		
						6B	11	0133F	269\$:	BRB	276\$	2697	
						00	ED	01341	270\$:	CMPZV	#0, #16, (R9), NO_DIGITS	2752	
2C	AE	69		10		15	18	01347		BGEQ	271\$		
						00	DD	01349		PUSHL	DBG\$GL_OPCODE_NAME		
						01	DD	0134F		PUSHL	#1		
				00028A3A		8F	DD	01351		PUSHL	#166458		
		00000000G	00			03	FB	01357		CALLS	#3, LIB\$SIGNAL		
FF7C	CD	69		60	AE4A	2C	AE	09	0135E	271\$:	CVTSP	NO DIGITS, TEMP_BUF2[BUF_OFFSET], (R9), -	2753
											TEMP_BUF1		
				13		6B	91	01368		CMPB	(R11), #19	2755	



			09	12	0136B	BNEQ	272\$		
		50	00000000G	00	9E 0136D	MOVAB	LIB\$AB_CVTPT_0, R0		
				07	11 01374	BRB	273\$		
		50	00000000G	00	9E 01376	MOVAB	LIB\$AB_CVTPT_2, R0		
69		60	FF7C	CD	24 0137D	CVTPT	(R9), TEMP_BUF1, (R0), (R9), @OUTPUT	2756	
				34	BE 01384				
				24	11 01386	BRB	276\$		2697
		1F	2C	AE	D1 01388	CMPL	NO DIGITS, #31		2761
				15	15 0138C	BLEQ	275\$		
			00000000G	00	DD 0138E	PUSHL	DBG\$GL_OPCODE_NAME		
				01	DD 01394	PUSHL	#1		
			00028A3A	8F	DD 01396	PUSHL	#166458		
				03	FB 0139C	CALLS	#3, LIB\$SIGNAL		
34	BE	69	00000000G	00	09 013A3	CVTSP	NO DIGITS, TEMP_BUF2[BUF_OFFSET], (R9), -	2762	
			60	AE4A			@OUTPUT		
				1C29	31 013AC	BRW	649\$		2187
			02	6B	91 013AF	CMPL	(R11), #2		2784
				12	13 013B2	BEQL	280\$		
			0E	6B	91 013B4	CMPL	(R11), #14		
				0D	13 013B7	BEQL	280\$		
			25	6B	91 013B9	CMPL	(R11), #37		
				03	1E 013BC	BGEQU	279\$		
				035C	31 013BE	BRW	327\$		
			27	6B	91 013C1	CMPL	(R11), #39		
				F8	1A 013C4	BGTRU	278\$		
		58	AE	32	B0 013C6	MOVW	#50, CLASS_S_DESC		2786
		5C	AE	AE	9E 013CA	MOVAB	TEMP_BUF2, -CLASS_S_DESC+4		2787
			60	57	D4 013CF	CLRL	DIGITS_IN_FRACT		2800
				52	D4 013D1	CLRL	R2		2801
				58	D5 013D3	TSTL	BIN SCALE		
				1C	13 013D5	BEQL	281\$		
				52	D6 013D7	INCL	R2		
			30	AE	D5 013D9	TSTL	SCALE		
				15	13 013DC	BEQL	281\$		
			00000000'	EF	9F 013DE	PUSHAB	P.AGX		2803
				01	DD 013E4	PUSHL	#1		
			00028362	8F	DD 013E6	PUSHL	#164706		
		00000000G	00	03	FB 013EC	CALLS	#3, LIB\$SIGNAL		
				58	D5 013F3	TSTL	BIN SCALE		2804
				03	18 013F5	BGEQ	282\$		
			57	58	CE 013F7	MNEGL	BIN SCALE, DIGITS_IN_FRACT		2806
				30	AE D5 013FA	TSTL	SCALE		2807
				04	18 013FD	BGEQ	283\$		
			57	30	AE CE 013FF	MNEGL	SCALE, DIGITS_IN_FRACT		2809
			06	56	D1 01403	CMPL	R6, #6		2814
				4E	12 01406	BNEQ	287\$		
		FF7C	CD	B0	AD 6E 01408	CVTLD	INTMED DATA, TEMP_BUF1		2816
				58	D5 0140E	TSTL	BIN SCALE		2821
				16	18 01410	BGEQ	285\$		
			51	FF7C	CD 9E 01412	MOVAB	TEMP_BUF1, R1		2823
			50	00000000'	EF 9E 01417	MOVAB	P.AGX, R0		
				00000000G	00 16 0141E	JSB	DBG\$CVT_DIVD2_R1		
				58	D6 01424	INCL	BIN SCALE		2824
				E6	11 01426	BRB	284\$		2821
				16	15 01428	BLEQ	286\$		2826
			51	FF7C	CD 9E 0142A	MOVAB	TEMP_BUF1, R1		2828
			50	00000000'	EF 9E 0142F	MOVAB	P.AGX, R0		

9  
)DBGCVTDX  
V04-000

M 12

15-Sep-1984 23:57:30

VAX-11 Bliss-32 V4.0-742

Page 132

14-Sep-1984 12:16:44

[DEBUG.SRC]DBGCVTDX.B32;1

(29)

			00000000G	00	16	01436	JSB	DBG\$CVT_MULD2_R1		
				58	D7	0143C	DECL	BIN_SCALE		2829
				E8	11	0143E	BRB	285\$		2826
			30	AE	DD	01440	286\$:	PUSHL	SCALE	2832
				57	DD	01443		PUSHL	DIGITS_IN_FRACT	
			60	AE	9F	01445		PUSHAB	CLASS_S_DESC	
			FF7C	CD	9F	01448		PUSHAB	TEMP_BUF1	
				04	FB	0144C		CALLS	#4, FOR\$CVT_D_TF	
				015A	31	01453		BRW	303\$	
			0C		56	D1	287\$:	CMPL	R6, #12	2835
				03	13	01459		BEQL	288\$	
				00FC	31	0145B		BRW	300\$	
			1A	04	BE	91	288\$:	CMPB	@4(SP), #26	2836
				4E	13	01462		BEQL	292\$	
			51	FF7C	CD	9E		MOVAB	TEMP_BUF1, R1	2839
			50	B0	AD	9E		MOVAB	INTMED_DATA, R0	
			00000000G	00	16	0146D		JSB	DBG\$CVT_CVTR0UH_R1	
			06	FF	AD	E9		BLBC	SRC_INFO+7, 289\$	2840
			FF7D	CD	80	8F		BISB2	#128, TEMP_BUF1+1	
				58	D5	0147D	289\$:	TSTL	BIN_SCALE	2846
				16	18	0147F		BGEQ	290\$	
			51	FF7C	CD	9E		MOVAB	TEMP_BUF1, R1	2848
			50	00000000'	EF	9E		MOVAB	P.AHA, R0	
			00000000G	00	16	0148D		JSB	DBG\$CVT_DIVH2_R1	
				58	D6	01493		INCL	BIN_SCALE	2849
				E6	11	01495		BRB	289\$	2846
				03	14	01497	290\$:	BGTR	291\$	2851
				0101	31	01499		BRW	302\$	
			51	FF7C	CD	9E	291\$:	MOVAB	TEMP_BUF1, R1	2853
			50	00000000'	EF	9E		MOVAB	P.AHB, R0	
			00000000G	00	16	014A8		JSB	DBG\$CVT_MULH2_R1	
				58	D7	014AE		DECL	BIN_SCALE	2854
				E5	11	014B0		BRB	290\$	2851
			15		52	E9	292\$:	BLBC	R2, 293\$	2869
			00000000'	EF	9F	014B5		PUSHAB	P.AHC	2871
				01	DD	014BB		PUSHL	#1	
			00028362	8F	DD	014BD		PUSHL	#164706	
			00000000G	00	03	FB		CALLS	#3, LIB\$SIGNAL	
				58	AE	B4	293\$:	CLRW	CLASS_S_DESC	2873
				10	28	014CD		MOV3	#16, INTMED_DATA, PREVIOUS_VALUE	2880
48	AE	B0	AD		10	28	294\$:	MOV3	#16, INTMED_DATA, PREVIOUS_VALUE	2900
48	AE	B0	AD		10	28		MOV3	INTMED_DATA, R0	2905
		B0	AD		9E	014D9		MOVAB	INTMED_DATA, R0	
		00000000G	00	16	014DD			JSB	DBG\$CVT_SCALE_OU_DOWN_BY_10_R1	
38	AE	B0	AD		10	28		MOV3	#16, INTMED_DATA, SAVED_VALUE	2912
		B0	AD		9E	014E9		MOVAB	INTMED_DATA, R0	2917
		00000000G	00	16	014ED			JSB	DBG\$CVT_SCALE_OU_UP_BY_10_R1	
			51	58	AE	3C		MOVZWL	CLASS_S_DESC, -CURRENT_POSITION	2925
				09	11	014F7		BRB	296\$	
		50	51	5C	AE	C1	295\$:	ADDL3	CLASS_S_DESC+4, CURRENT_POSITION, R0	
			A0	60	90	014FE		MOV3	(R0), -17(R0)	
		01	F4	51	F4	01502	296\$:	SUBGEQ	CURRENT_POSITION, 295\$	
		50	AE	B0	AD	C3		SUBL3	INTMED_DATA, PREVIOUS_VALUE, R0	2930
5C	BE	48	50		30	81		ADDB3	#48, R0, @CLASS_S_DESC+4	
				58	AE	B6		INCW	CLASS_S_DESC	2937
		38	AE		10	28		MOV3	#16, SAVED_VALUE, INTMED_DATA	2944
B0	AD			BC	AD	D5		TSTL	INTMED_DATA+12	2947
				B5	12	0151C		BNEQ	294\$	

PC	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

			57	D5	015E6	308\$:	TSTL	DIGITS_IN_FRACT	3001
			02	12	015E8		BNEQ	309\$	3003
			55	D7	015EA		DECL	FINAL_LEN	3005
	58	72	6E	E8	015EC	309\$:	BLBS	STATUS, 317\$	3008
		AE	32	B0	015EF		MOVW	#50, CLASS_S_DESC	3009
		1E	56	D1	015F3		CMPL	R6, #30	3011
			05	12	015F6		BNEQ	310\$	3013
		57	1F	D0	015F8		MOVL	#31, DIGITS_IN_FRACT	3015
			1D	11	015FB		BRB	313\$	3017
		09	AD	B1	015FD	310\$:	CMPL	DST_INFO+5, #9	3018
			05	1A	01601		BGTRU	311\$	3019
		57	21	D0	01603		MOVL	#33, DIGITS_IN_FRACT	3020
			12	11	01606		BRB	313\$	3021
		50	AD	3C	01608	311\$:	MOVZWL	DST_INFO+5, R0	3022
			09	C2	0160C		SUBL2	#9, R0	3023
		50	50	D1	0160F		CMPL	R0, #33	3024
		21	03	15	01612		BLEQ	312\$	3025
		50	21	D0	01614		MOVL	#33, R0	3026
		57	50	D0	01617	312\$:	MOVL	R0, DIGITS_IN_FRACT	3027
			04	DD	0161A	313\$:	PUSHL	#4	3028
			7E	D4	0161C		CLRL	-(SP)	3029
		38	AE	DD	0161E		PUSHL	SCALE	3030
			57	DD	01621		PUSHL	DIGITS_IN_FRACT	3031
		68	AE	9F	01623		PUSHAB	CLASS_S_DESC	3032
		FF7C	CD	9F	01626		PUSHAB	TEMP_BUF1	3033
		00000000G	00	06	FB	0162A	CALLS	#6, FOR\$CVT_H_TE	3034
			6E	D0	01631		MOVL	R0, STATUS	3035
			15	6E	E8	01634	BLBS	STATUS, 314\$	3036
				EF	9F	01637	PUSHAB	P.AHE	3037
				01	DD	0163D	PUSHL	#1	3038
				8F	DD	0163F	PUSHL	#164706	3039
		00000000G	00	03	FB	01645	CALLS	#3, LIB\$SIGNAL	3040
60	AE		32	20	3B	0164C	314\$:	SKPC	#32, #50, TEMP_BUF2
				02	12	01651		BNEQ	316\$
				51	D4	01653	315\$:	CLRL	R1
			50	AE	9E	01655	316\$:	MOVAB	TEMP_BUF2, R0
		5A	51	50	C3	01659		SUBL3	R0, R1, BUF_OFFSET
		55	32	5A	C3	0165D		SUBL3	BUF_OFFSET, #50, FINAL_LEN
			14	55	D0	01661	317\$:	MOVL	FINAL_LEN, OUTPUT_STR_LEN
			26	6B	91	01665		CMPL	(R11), #38
				46	12	01668		BNEQ	321\$
		58	AE	55	B0	0166A		MOVW	FINAL_LEN, CLASS_S_DESC
5C	AE	34	AE	01	C1	0166E		ADDL3	#1, OUTPUT, CLASS_S_DESC+4
			52	AE	9E	01674		MOVAB	CLASS_S_DESC, R2
			51	60	AE	9E	01678	MOVAB	TEMP_BUF2[BUF_OFFSET], R1
			50	55	D0	0167D		MOVL	FINAL_LEN, R0
				00	16	01680		JSB	LIB\$COPY_R_DX6
		00000000G	6E	50	D0	01686		MOVL	R0, STATUS
			8F	6E	D1	01689		CMPL	STATUS, #LIB\$_STRTRU
				15	12	01690		BNEQ	319\$
				00	DD	01692	318\$:	PUSHL	DBG\$GL_OPCODE_NAME
				01	DD	01698		PUSHL	#1
				8F	DD	0169A		PUSHL	#165547
		00000000G	00	03	FB	016A0		CALLS	#3, LIB\$SIGNAL
			03	6E	E9	016A7	319\$:	BLBC	STATUS, 320\$
				1761	31	016AA		BRW	622\$
				1755	31	016AD	320\$:	BRW	621\$

27		6B	91	016B0	321\$:	CMPB	(R11), #39	3039
		45	12	016B3		BNEQ	325\$	
58	AE	55	B0	016B5		MOVW	FINAL_LEN, CLASS_S_DESC	3043
5C	AE		D0	016B9		MOVL	OUTPUT, CLASS_S_DESC+4	3044
52		34	AE	016BE		MOVAB	CLASS_S_DESC, R2	3045
51		58	AE	016C2		MOVAB	TEMP_BUF2[BUF_OFFSET], R1	
50		60	AE4A	016C7		MOVL	FINAL_LEN, R0	
	00000000G		55	D0	016C7	JSB	LIB\$COPY_R_DX6	
			00	16	016CA	MOVL	R0, STATUS	
00000000G	6E		50	D0	016D0	CMPL	STATUS, #LIB\$_STRTRU	3046
	8F		6E	D1	016D3	BNEQ	323\$	
			15	12	016DA	PUSHL	DBG\$GL_OPCODE_NAME	
	00000000G		00	DD	016DC	PUSHL	#1	
			01	DD	016E2	PUSHL	#165547	
00000000G		000286AB	8F	DD	016E4	CALLS	#3, LIB\$SIGNAL	
	00		03	FB	016EA	BLBC	STATUS, 324\$	3047
	03		6E	E9	016F1	BRW	627\$	
			176D	31	016F4	BRW	626\$	
			1761	31	016F7	MOVAB	TEMP_BUF2[BUF_OFFSET], R1	3053
51		60	AE4A	9E	016FA	MOVL	R9, R2	
52			59	D0	016FF	MOVL	FINAL_LEN, R0	
50			55	D0	01702	JSB	LIB\$COPY_R_DX6	
	00000000G		00	16	01705	MOVL	R0, STATUS	
00000000G	6E		50	D0	0170B	CMPL	STATUS, #LIB\$_STRTRU	3054
	8F		6E	D1	0170E	BEQL	326\$	
			03	13	01715	BRW	641\$	
			1840	31	01717	BRW	640\$	
			1828	31	0171A	CMPL	R6, #6	3063
06			56	D1	0171D	BNEQ	328\$	
	00000000'		08	12	01720	PUSHAB	P.AHF	3064
			EF	9F	01722	BRB	331\$	
			1B	11	01728	CMPL	R6, #12	3065
0C			56	D1	0172A	BNEQ	329\$	
	00000000'		08	12	0172D	PUSHAB	P.AHG	3066
			EF	9F	0172F	BRB	331\$	
			0E	11	01735	CMPL	R6, #30	3067
1E			56	D1	01737	BEQL	330\$	
			03	13	0173A	BRW	649\$	
			1899	31	0173C	PUSHAB	P.AHH	3068
	00000000'		EF	9F	0173F	BRW	647\$	
			1881	31	01745	TSTL	BIN_SCALE	3074
			58	D5	01748	BLEQ	333\$	
			26	15	0174A	MOVAB	INTMED_DATA, R1	
51		B0	AD	9E	0174C	MOVAB	P.AHI, R0	
50	00000000'		EF	9E	01750	JSB	DBG\$CVT_MULD2_R1	
	00000000G		00	16	01757	MOVAB	INTMED_DATA+8, R1	
			AD	9E	0175D	MOVAB	P.AHJ, R0	
51	00000000'	B8	AD	9E	01761	JSB	DBG\$CVT_MULD2_R1	
50	00000000'		EF	9E	01768	DECL	BIN_SCALE	
	00000000G		00	16	0176E	BRB	332\$	
			58	D7	01770	BGEQ	334\$	
			D6	11	01772	MOVAB	INTMED_DATA, R1	
51		B0	AD	9E	01774	MOVAB	P.AHK, R0	
50	00000000'		EF	9E	01778	JSB	DBG\$CVT_DIVD2_R1	
	00000000G		00	16	0177F	MOVAB	INTMED_DATA+8, R1	
			AD	9E	01785	MOVAB	P.AHL, R0	
51	00000000'	B8	AD	9E	01789	JSB	DBG\$CVT_DIVD2_R1	
50	00000000'		EF	9E	01790			
	00000000G		00	16	01790			



			00000000'	EF	9F	01859	340\$:	PUSHAB	P.AHQ	3123
				1767	31	0185F		BRW	647\$	
	52	F5		AD	3C	01862	341\$:	MOVZWL	DST_INFO+5, R2	3116
	50			01	CE	01866		MNEGL	#1, I	3118
				OC	11	01869		BRB	343\$	
	01			50	EF	0186B	342\$:	EXTZV	I, #1, INTMED_DATA, R1	
	50			51	FO	01871		INSV	R1, I, #1, @OUTPUT	
	50			52	F2	01877	343\$:	AOBLSS	R2, I, 342\$	3116
			175A	31	0187B			BRW	649\$	2187
				58	D5	0187E	344\$:	TSTL	BIN SCALE	3128
				26	15	01880		BLEQ	345\$	
	51	B0		AD	9E	01882		MOVAB	INTMED_DATA, R1	
	50	00000000'		EF	9E	01886		MOVAB	P.AHR,-R0	
		00000000G		00	16	0188D		JSB	DBG\$CVT_MULD2_R1	
	51	B8		AD	9E	01893		MOVAB	INTMED_DATA+8, R1	
	50	00000000'		EF	9E	01897		MOVAB	P.AHS,-R0	
		00000000G		00	16	0189E		JSB	DBG\$CVT_MULD2_R1	
				58	D7	018A4		DECL	BIN SCALE	
				D6	11	018A6		BRB	344\$	
				26	18	018A8	345\$:	BGEQ	346\$	
	51	B0		AD	9E	018AA		MOVAB	INTMED_DATA, R1	
	50	00000000'		EF	9E	018AE		MOVAB	P.AHT,-R0	
		00000000G		00	16	018B5		JSB	DBG\$CVT_DIVD2_R1	
	51	B8		AD	9E	018BB		MOVAB	INTMED_DATA+8, R1	
	50	00000000'		EF	9E	018BF		MOVAB	P.AHU,-R0	
		00000000G		00	16	018C6		JSB	DBG\$CVT_DIVD2_R1	
				58	D6	018CC		INCL	BIN SCALE	
				D8	11	018CE		BRB	345\$	
			30	AE	D5	018D0	346\$:	TSTL	SCALE	
				27	15	018D3		BLEQ	347\$	
	51	B0		AD	9E	018D5		MOVAB	INTMED_DATA, R1	
	50	00000000'		EF	9E	018D9		MOVAB	P.AHV,-R0	
		00000000G		00	16	018E0		JSB	DBG\$CVT_MULD2_R1	
	51	B8		AD	9E	018E6		MOVAB	INTMED_DATA+8, R1	
	50	00000000'		EF	9E	018EA		MOVAB	P.AHW,-R0	
		00000000G		00	16	018F1		JSB	DBG\$CVT_MULD2_R1	
			30	AE	D7	018F7		DECL	SCALE	
				D4	11	018FA		BRB	346\$	
				27	18	018FC	347\$:	BGEQ	348\$	
	51	B0		AD	9E	018FE		MOVAB	INTMED_DATA, R1	
	50	00000000'		EF	9E	01902		MOVAB	P.AHX,-R0	
		00000000G		00	16	01909		JSB	DBG\$CVT_DIVD2_R1	
	51	B8		AD	9E	0190F		MOVAB	INTMED_DATA+8, R1	
	50	00000000'		EF	9E	01913		MOVAB	P.AHY,-R0	
		00000000G		00	16	0191A		JSB	DBG\$CVT_DIVD2_R1	
			30	AE	D6	01920		INCL	SCALE	

Address	Instruction	Comment	Symbol
00000000	BRB	347\$	
00000001	CASEB	(R11), #4, #22	
00000002	.WORD	351\$-349\$,-	
00000003		355\$-349\$,-	
00000004		350\$-349\$,-	
00000005		350\$-349\$,-	
00000006		350\$-349\$,-	
00000007		350\$-349\$,-	
00000008		350\$-349\$,-	
00000009		350\$-349\$,-	
0000000A		350\$-349\$,-	
0000000B		350\$-349\$,-	
0000000C		350\$-349\$,-	
0000000D		350\$-349\$,-	
0000000E		350\$-349\$,-	
0000000F		350\$-349\$,-	
00000010		350\$-349\$,-	
00000011		350\$-349\$,-	
00000012		350\$-349\$,-	
00000013		350\$-349\$,-	
00000014		350\$-349\$,-	
00000015		350\$-349\$,-	
00000016		350\$-349\$,-	
00000017		350\$-349\$,-	
00000018		350\$-349\$,-	
00000019		350\$-349\$,-	
0000001A		350\$-349\$,-	
0000001B		350\$-349\$,-	
0000001C		350\$-349\$,-	
0000001D		350\$-349\$,-	
0000001E		350\$-349\$,-	
0000001F		350\$-349\$,-	
00000020		350\$-349\$,-	
00000021		350\$-349\$,-	
00000022		350\$-349\$,-	
00000023		350\$-349\$,-	
00000024		350\$-349\$,-	
00000025		350\$-349\$,-	
00000026		350\$-349\$,-	
00000027		350\$-349\$,-	
00000028		350\$-349\$,-	
00000029		350\$-349\$,-	
0000002A		350\$-349\$,-	
0000002B		350\$-349\$,-	
0000002C		350\$-349\$,-	
0000002D		350\$-349\$,-	
0000002E		350\$-349\$,-	
0000002F		350\$-349\$,-	
00000030		350\$-349\$,-	
00000031		350\$-349\$,-	
00000032		350\$-349\$,-	
00000033		350\$-349\$,-	
00000034		350\$-349\$,-	
00000035		350\$-349\$,-	
00000036		350\$-349\$,-	
00000037		350\$-349\$,-	
00000038		350\$-349\$,-	
00000039		350\$-349\$,-	
0000003A		350\$-349\$,-	
0000003B		350\$-349\$,-	
0000003C		350\$-349\$,-	
0000003D		350\$-349\$,-	
0000003E		350\$-349\$,-	
0000003F		350\$-349\$,-	
00000040		350\$-349\$,-	
00000041		350\$-349\$,-	
00000042		350\$-349\$,-	
00000043		350\$-349\$,-	
00000044		350\$-349\$,-	
00000045		350\$-349\$,-	
00000046		350\$-349\$,-	
00000047		350\$-349\$,-	
00000048		350\$-349\$,-	
00000049		350\$-349\$,-	
0000004A		350\$-349\$,-	
0000004B		350\$-349\$,-	
0000004C		350\$-349\$,-	
0000004D		350\$-349\$,-	
0000004E		350\$-349\$,-	
0000004F		350\$-349\$,-	
00000050		350\$-349\$,-	
00000051		350\$-349\$,-	
00000052		350\$-349\$,-	
00000053		350\$-349\$,-	
00000054		350\$-349\$,-	
00000055		350\$-349\$,-	



01	58 5C	AE AE 0A 0009	60 04	71 72 AE BE 0004	1A 019D5 B0 019D7 358\$: 9E 019DB 8F 019E0 019E5 359\$:	BGTRU 367\$ MOVW #50, CLASS_S_DESC MOVAB TEMP_BUF2, CLASS_S_DESC+4 CASEB @4(SP), #10, #1 .WORD 360\$-359\$,- 361\$-359\$	3166 3167 3170
		57		07	D0 019E9 360\$:	MOVL #7, DIGITS_IN_FRACT	
		57		03	11 019EC	BRB 362\$	
		07	F5	10	D0 019EE 361\$:	MOVL #16, DIGITS_IN_FRACT	
				AD	B1 019F1 362\$:	CMPW DST_INFO+5, #7	3180
		51	F5	15	1B 019F5	BLEQU 364\$	
		51		AD	3C 019F7	MOVZWL DST_INFO+5, R1	3183
		50		07	C2 019FB	SUBL2 #7, R1	
		51		57	D0 019FE	MOVL DIGITS_IN_FRACT, R0	
				50	D1 01A01	CML R0, R1	
				03	15 01A04	BLEQ 363\$	
		50		51	D0 01A06	MOVL R1, R0	
		57		50	D0 01A09 363\$:	MOVL R0, DIGITS_IN_FRACT	3182
				7E	D4 01A0C 364\$:	CLRL -(SP)	3184
			34	AE	DD 01A0E	PUSHL SCALE	
				57	DD 01A11	PUSHL DIGITS_IN_FRACT	
			64	AE	9F 01A13	PUSHAB CLASS_S_DESC	
			80	AD	9F 01A16	PUSHAB INTMED_DATA	
	00000000G	00		05	FB 01A19	CALLS #5, FOR\$CVT_D_TE	
		6E		50	D0 01A20	MOVL R0, STATUS	
		15		6E	E8 01A23	BLBS STATUS, 365\$	3185
				00000000'	EF 9F 01A26	PUSHAB P.AIA	
				00028362	01 DD 01A2C	PUSHL #1	
					8F DD 01A2E	PUSHL #164706	
					03 FB 01A34	CALLS #3, LIB\$SIGNAL	
60	AE	00000000G	00	20	3B 01A3B 365\$:	SKPC #32, #50, TEMP_BUF2	3186
			32	03	13 01A40	BEQL 366\$	
				FC10	31 01A42	BRW 316\$	
				FC0B	31 01A45 366\$:	BRW 315\$	
				00000000'	EF 9F 01A48 367\$:	PUSHAB P.AIB	3226
				1578	31 01A4E	BRW 647\$	
		1B	04	BE	91 01A51 368\$:	CMPB @4(SP), #27	3231
				09	13 01A55	BEQL 369\$	
		1D	04	BE	91 01A57	CMPB @4(SP), #29	3232
				03	13 01A5B	BEQL 369\$	
				00CB	31 01A5D	BRW 374\$	
		51	FF7C	CD	9E 01A60 369\$:	MOVAB TEMP_BUF1, R1	3233
		50	B0	AD	9E 01A65	MOVAB INTMED_DATA, R0	
			00000000G	00	16 01A69	JSB DBG\$CVT_CVTGH_R1	
		51	C0	AD	9E 01A6F	MOVAB INTMED_DATA+16, R1	
		50	B8	AD	9E 01A73	MOVAB INTMED_DATA+8, R0	
			00000000G	00	16 01A77	JSB DBG\$CVT_CVTGH_R1	
80	AD	FF7C	CD	10	28 01A7D 370\$:	MOV3 #16, TEMP_BUF1, INTMED_DATA	
				58	D5 01A84	TSTL BIN_SCALE	
				26	15 01A86	BLEQ 371\$	
		51	B0	AD	9E 01A88	MOVAB INTMED_DATA, R1	
		50	00000000'	EF	9E 01A8C	MOVAB P.AIC, -R0	
			00000000G	00	16 01A93	JSB DBG\$CVT_MULH2_R1	
		51	C0	AD	9E 01A99	MOVAB INTMED_DATA+16, R1	
		50	00000000'	EF	9E 01A9D	MOVAB P.AID, -R0	
			00000000G	00	16 01AA4	JSB DBG\$CVT_MULH2_R1	
				58	D7 01AAA	DECL BIN_SCALE	

	D6	11	01AAC		BRB	370\$
	26	18	01AAE	371\$:	BGEQ	372\$
51	AD	9E	01AB0		MOVAB	INTMED_DATA, R1
50	EF	9E	01AB4		MOVAB	P.AIE,-R0
	00	16	01ABB		JSB	DBG\$CVT_DIVH2_R1
51	AD	9E	01AC1		MOVAB	INTMED_DATA+16, R1
50	EF	9E	01AC5		MOVAB	P.AIF,-R0
	00	16	01ACC		JSB	DBG\$CVT_DIVH2_R1
	58	D6	01AD2		INCL	BIN_SCALE
	D8	11	01AD4		BRB	371\$
	30	AE	D5 01AD6	372\$:	TSTL	SCALE
	27	15	01AD9		BLEQ	373\$
51	AD	9E	01ADB		MOVAB	INTMED_DATA, R1
50	EF	9E	01ADF		MOVAB	P.AIG,-R0
	00	16	01AE6		JSB	DBG\$CVT_MULH2_R1
51	AD	9E	01AEC		MOVAB	INTMED_DATA+16, R1
50	EF	9E	01AF0		MOVAB	P.AIH,-R0
	00	16	01AF7		JSB	DBG\$CVT_MULH2_R1
	30	AE	D7 01AFD		DECL	SCALE
	D4	11	01B00		BRB	372\$
	51	18	01B02	373\$:	BGEQ	375\$
51	AD	9E	01B04		MOVAB	INTMED_DATA, R1
50	EF	9E	01B08		MOVAB	P.AII,-R0
	00	16	01B0F		JSB	DBG\$CVT_DIVH2_R1
51	AD	9E	01B15		MOVAB	INTMED_DATA+16, R1
50	EF	9E	01B19		MOVAB	P.AIJ,-R0
	00	16	01B20		JSB	DBG\$CVT_DIVH2_R1
	30	AE	D6 01B26		INCL	SCALE
	D7	11	01B29		BRB	373\$
	58	D5	01B2B	374\$:	TSTL	BIN_SCALE
	26	15	01B2D		BLEQ	375\$
51	AD	9E	01B2F		MOVAB	INTMED_DATA, R1
50	EF	9E	01B33		MOVAB	P.AIK,-R0
	00	16	01B3A		JSB	DBG\$CVT_MULH2_R1
51	AD	9E	01B40		MOVAB	INTMED_DATA+16, R1
50	EF	9E	01B44		MOVAB	P.AIL,-R0
	00	16	01B4B		JSB	DBG\$CVT_MULH2_R1
	58	D7	01B51		DECL	BIN_SCALE
	D6	11	01B53		BRB	374\$
	58	D5	01B55	375\$:	TSTL	BIN_SCALE
	26	18	01B57		BGEQ	376\$
51	AD	9E	01B59		MOVAB	INTMED_DATA, R1
50	EF	9E	01B5D		MOVAB	P.AIM,-R0
	00	16	01B64		JSB	DBG\$CVT_DIVH2_R1
51	AD	9E	01B6A		MOVAB	INTMED_DATA+16, R1
50	EF	9E	01B6E		MOVAB	P.AIN,-R0
	00	16	01B75		JSB	DBG\$CVT_DIVH2_R1
	58	D6	01B7B		INCL	BIN_SCALE
	D6	11	01B7D		BRB	375\$
	30	AE	D5 01B7F	376\$:	TSTL	SCALE
	27	15	01B82		BLEQ	377\$
51	AD	9E	01B84		MOVAB	INTMED_DATA, R1
50	EF	9E	01B88		MOVAB	P.AIO,-R0
	00	16	01B8F		JSB	DBG\$CVT_MULH2_R1
51	AD	9E	01B95		MOVAB	INTMED_DATA+16, R1
50	EF	9E	01B99		MOVAB	P.AIP,-R0
	00	16	01BA0		JSB	DBG\$CVT_MULH2_R1

[illegible]

Address	Hex	Assembly	Comment
000000FF	8F FF7C	PUSHAB	382\$: 387\$-381\$,-
0000000G	00 C00286A3	BRW	383\$: 387\$-381\$,-
0000000G	00 0CEA	CMPL	384\$: 387\$-381\$
0000FFFF	8F FF7C	BLEQU	385\$: P.AIS
0000000G	00 0000000G	PUSHL	386\$: 647\$
0000000G	00 000286A3	PUSHL	387\$: TEMP_BUF1, #255
0000000G	00 0D3C	PUSHL	388\$: DBG\$GL_OPCODE_NAME
0000000G	52 F5	CALLS	389\$: #1
0000000G	50 01	BRW	390\$: #3, LIB\$SIGNAL
0000000G	01 50	MOVZWL	391\$: 549\$
0000000G	50 52	MNEGL	392\$: TEMP_BUF1, #65535
0000000G	1B 04	BRB	393\$: 386\$
0000000G	1D 04	EXTZV	394\$: DBG\$GL_OPCODE_NAME
0000000G	51 FF7C	INSV	395\$: #165539
0000000G	50 B0	AOBLS	396\$: #3, LIB\$SIGNAL
0000000G	51 C0	BRW	397\$: 560\$
0000000G	50 B8	MOVZWL	398\$: DST_INFO+5, R2
0000000G	CD 10	MNEGL	399\$: #1, I
0000000G	51 58	BRB	400\$: 389\$
0000000G	50 26	EXTZV	401\$: I, #1, INTMED_DATA, R1
0000000G	51 B0	INSV	402\$: R1, 1, #1, @OUTPUT
0000000G	50 00	AOBLS	403\$: R2, 1, 388\$
0000000G	51 C0	BRW	404\$: 649\$
0000000G	50 B8	MOVZWL	405\$: @4(SP), #27
0000000G	CD 10	MNEGL	406\$: 391\$
0000000G	51 58	BRB	407\$: @4(SP), #29
0000000G	50 26	EXTZV	408\$: 391\$
0000000G	51 B0	INSV	409\$: 396\$
0000000G	50 00	AOBLS	410\$: TEMP_BUF1, R1
0000000G	51 C0	BRW	411\$: INTMED_DATA, R0
0000000G	50 B8	MOVZWL	412\$: DBG\$CVT_CVTGH R1
0000000G	CD 10	MNEGL	413\$: INTMED_DATA+16, R1
0000000G	51 58	BRB	414\$: INTMED_DATA+8, R0
0000000G	50 26	EXTZV	415\$: DBG\$CVT_CVTGH R1
0000000G	51 B0	INSV	416\$: #16, TEMP_BUF1, INTMED_DATA
0000000G	50 00	AOBLS	417\$: BIN_SCALE
0000000G	51 C0	BRW	418\$: 393\$
0000000G	50 B8	MOVZWL	419\$: INTMED_DATA, R1
0000000G	CD 10	MNEGL	420\$: P.AIT, R0
0000000G	51 58	BRB	421\$: DBG\$CVT_MULH2 R1
0000000G	50 26	EXTZV	422\$: INTMED_DATA+16, R1
0000000G	51 B0	INSV	423\$: P.AIU, R0
0000000G	50 00	AOBLS	424\$: DBG\$CVT_MULH2 R1
0000000G	51 C0	BRW	425\$: BIN_SCALE
0000000G	50 B8	MOVZWL	426\$: 392\$
0000000G	CD 10	MNEGL	427\$: 394\$
0000000G	51 58	BRB	428\$: INTMED_DATA, R1
0000000G	50 26	EXTZV	429\$: P.AIV, R0
0000000G	51 B0	INSV	430\$: DBG\$CVT_DIVH2 R1
0000000G	50 00	AOBLS	431\$: INTMED_DATA+16, R1
0000000G	51 C0	BRW	432\$: P.AIW, R0
0000000G	50 B8	MOVZWL	433\$: DBG\$CVT_DIVH2 R1
0000000G	CD 10	MNEGL	434\$: BIN_SCALE
0000000G	51 58	BRB	435\$: 393\$
0000000G	50 26	EXTZV	436\$: 394\$
0000000G	51 B0	INSV	437\$: INTMED_DATA, R1
0000000G	50 00	AOBLS	438\$: P.AIV, R0
0000000G	51 C0	BRW	439\$: DBG\$CVT_DIVH2 R1
0000000G	50 B8	MOVZWL	440\$: INTMED_DATA+16, R1
0000000G	CD 10	MNEGL	441\$: P.AIW, R0
0000000G	51 58	BRB	442\$: DBG\$CVT_DIVH2 R1
0000000G	50 26	EXTZV	443\$: BIN_SCALE
0000000G	51 B0	INSV	444\$: 393\$
0000000G	50 00	AOBLS	445\$: 394\$
0000000G	51 C0	BRW	446\$: INTMED_DATA, R1
0000000G	50 B8	MOVZWL	447\$: P.AIV, R0
0000000G	CD 10	MNEGL	448\$: DBG\$CVT_DIVH2 R1
0000000G	51 58	BRB	449\$: INTMED_DATA+16, R1
0000000G	50 26	EXTZV	450\$: P.AIW, R0
0000000G	51 B0	INSV	451\$: DBG\$CVT_DIVH2 R1
0000000G	50 00	AOBLS	452\$: BIN_SCALE
0000000G</			

```

30 AE D5 01D38 394$: TSTL SCALE
27 15 01D38 BLEQ 395$
51 B0 AD 9E 01D3D MOVAB INTMED_DATA, R1
50 00000000' EF 9E 01D41 MOVAB P.AIX, R0
00000000G 00 16 01D48 JSB DBG$CVT_MULH2_R1
51 C0 AD 9E 01D4E MOVAB INTMED_DATA+16, R1
50 00000000' EF 9E 01D52 MOVAB P.AIY, R0
00000000G 00 16 01D59 JSB DBG$CVT_MULH2_R1
30 AE D7 01D5F DECL SCALE
D4 11 01D62 BRB 394$
51 18 01D64 395$: BGEQ 397$
51 B0 AD 9E 01D66 MOVAB INTMED_DATA, R1
50 00000000' EF 9E 01D6A MOVAB P.AIZ, R0
00000000G 00 16 01D71 JSB DBG$CVT_DIVH2_R1
51 C0 AD 9E 01D77 MOVAB INTMED_DATA+16, R1
50 00000000' EF 9E 01D7B MOVAB P.AJA, R0
00000000G 00 16 01D82 JSB DBG$CVT_DIVH2_R1
30 AE D6 01D88 INCL SCALE
D7 11 01D8B BRB 395$
58 D5 01D8D 396$: TSTL BIN SCALE
26 15 01D8F BLEQ 397$
51 B0 AD 9E 01D91 MOVAB INTMED_DATA, R1
50 00000000' EF 9E 01D95 MOVAB P.AJB, R0
00000000G 00 16 01D9C JSB DBG$CVT_MULH2_R1
51 C0 AD 9E 01DA2 MOVAB INTMED_DATA+16, R1
50 00000000' EF 9E 01DA6 MOVAB P.AJC, R0
00000000G 00 16 01DAD JSB DBG$CVT_MULH2_R1
58 D7 01DB3 DECL BIN SCALE
D6 11 01DB5 BRB 396$
58 D5 01DB7 397$: TSTL BIN SCALE
26 18 01DB9 BGEQ 398$
51 B0 AD 9E 01DBB MOVAB INTMED_DATA, R1
50 00000000' EF 9E 01DBF MOVAB P.AJD, R0
00000000G 00 16 01DC6 JSB DBG$CVT_DIVH2_R1
51 C0 AD 9E 01DCC MOVAB INTMED_DATA+16, R1
50 00000000' EF 9E 01DD0 MOVAB P.AJE, R0
00000000G 00 16 01DD7 JSB DBG$CVT_DIVH2_R1
58 D6 01DDD INCL BIN SCALE
D6 11 01DDF BRB 397$
30 AE D5 01DE1 398$: TSTL SCALE
27 15 01DE4 BLEQ 399$
51 B0 AD 9E 01DE6 MOVAB INTMED_DATA, R1
50 00000000' EF 9E 01DEA MOVAB P.AJF, R0
00000000G 00 16 01DF1 JSB DBG$CVT_MULH2_R1
51 C0 AD 9E 01DF7 MOVAB INTMED_DATA+16, R1
50 00000000' EF 9E 01DFB MOVAB P.AJG, R0
00000000G 00 16 01E02 JSB DBG$CVT_MULH2_R1
30 AE D7 01E08 DECL SCALE
D4 11 01E0B BRB 398$
27 18 01E0D 399$: BGEQ 400$
51 B0 AD 9E 01E0F MOVAB INTMED_DATA, R1
50 00000000' EF 9E 01E13 MOVAB P.AJH, R0
00000000G 00 16 01E1A JSB DBG$CVT_DIVH2_R1
51 C0 AD 9E 01E20 MOVAB INTMED_DATA+16, R1
50 00000000' EF 9E 01E24 MOVAB P.AJI, R0
00000000G 00 16 01E2B JSB DBG$CVT_DIVH2_R1
30 AE D6 01E31 INCL SCALE

```

3293

[illegible]

```

          51      CO      AD      9E 01EE9      MOVAB      INTMED_DATA+16, R1
          50      B8      AD      9E 01EED      MOVAB      INTMED_DATA+8, R0
          00000000G 00      16 01EF1      JSB          DBG$CVT_CVTGH_R1
BO      AD      FF7C  CD      10 28 01EF7      MOV3C3      #16, TEMP_BUFT, INTMED_DATA
          58 D5 01EFE 412$: TSTL      BIN_SCALE
          26 15 01F00      BLEQ      413$
          51      B0      AD      9E 01F02      MOVAB      INTMED_DATA, R1
          50 00000000' EF      9E 01F06      MOVAB      P.AJK, -R0
          00000000G 00      16 01F0D      JSB          DBG$CVT_MULH2_R1
          51      CO      AD      9E 01F13      MOVAB      INTMED_DATA+16, R1
          50 00000000' EF      9E 01F17      MOVAB      P.AJL, -R0
          00000000G 00      16 01F1E      JSB          DBG$CVT_MULH2_R1
          58 D7 01F24      DECL      BIN_SCALE
          D6 11 01F26      BRB          412$
          26 18 01F28 413$: BGEQ      414$
          51      B0      AD      9E 01F2A      MOVAB      INTMED_DATA, R1
          50 00000000' EF      9E 01F2E      MOVAB      P.AJM, -R0
          00000000G 00      16 01F35      JSB          DBG$CVT_DIVH2_R1
          51      CO      AD      9E 01F3B      MOVAB      INTMED_DATA+16, R1
          50 00000000' EF      9E 01F3F      MOVAB      P.AJN, -R0
          00000000G 00      16 01F46      JSB          DBG$CVT_DIVH2_R1
          58 D6 01F4C      INCL      BIN_SCALE
          D8 11 01F4E      BRB          413$
          30      AE      D5 01F50 414$: TSTL      SCALE
          27 15 01F53      BLEQ      415$
          51      B0      AD      9E 01F55      MOVAB      INTMED_DATA, R1
          50 00000000' EF      9E 01F59      MOVAB      P.AJO, -R0
          00000000G 00      16 01F60      JSB          DBG$CVT_MULH2_R1
          51      CO      AD      9E 01F66      MOVAB      INTMED_DATA+16, R1
          50 00000000' EF      9E 01F6A      MOVAB      P.AJP, -R0
          00000000G 00      16 01F71      JSB          DBG$CVT_MULH2_R1
          30      AE      D7 01F77      DECL      SCALE
          D4 11 01F7A      BRB          414$
          51      B0      AD      9E 01F7E 415$: BGEQ      417$
          50 00000000' EF      9E 01F82      MOVAB      INTMED_DATA, R1
          00000000G 00      16 01F89      MOVAB      P.AJQ, -R0
          51      CO      AD      9E 01F8F      JSB          DBG$CVT_DIVH2_R1
          50 00000000' EF      9E 01F93      MOVAB      INTMED_DATA+16, R1
          00000000G 00      16 01F9A      MOVAB      P.AJR, -R0
          30      AE      D6 01FA0      JSB          DBG$CVT_DIVH2_R1
          D7 11 01FA3      INCL      SCALE
          58 D5 01FA5 416$: BRB          415$
          26 15 01FA7      TSTL      BIN_SCALE
          51      B0      AD      9E 01FA9      BLEQ      417$
          50 00000000' EF      9E 01FAD      MOVAB      INTMED_DATA, R1
          00000000G 00      16 01FB4      MOVAB      P.AJS, -R0
          51      CO      AD      9E 01FBA      JSB          DBG$CVT_MULH2_R1
          50 00000000' EF      9E 01FBE      MOVAB      INTMED_DATA+16, R1
          00000000G 00      16 01FC5      MOVAB      P.AJT, -R0
          58 D7 01FCB      JSB          DBG$CVT_MULH2_R1
          D6 11 01FCD      DECL      BIN_SCALE
          58 D5 01FCF 417$: BRB          416$
          26 18 01FD1      TSTL      BIN_SCALE
          51      B0      AD      9E 01FD3      BLEQ      418$
          50 00000000' EF      9E 01FD7      MOVAB      INTMED_DATA, R1
          00000000G 00      16 01FDE      MOVAB      P.AJU, -R0
          JSB          DBG$CVT_DIVH2_R1

```

	51		C0	AD	9E	01FE4	MOVAB	INTMED_DATA+16, R1	
	50	00000000'		EF	9E	01FE8	MOVAB	P.AJV, R0	
		00000000G		00	16	01FEF	JSB	DBG\$CVT_DIVH2_R1	
				58	D6	01FF5	INCL	BIN_SCALE	
			30	D6	11	01FF7	BRB	417\$	
				AE	D5	01FF9	418\$:	TSTL	SCALE
				27	15	01FFC		BLEQ	419\$
	51		B0	AD	9E	01FFE	MOVAB	INTMED_DATA, R1	
	50	00000000'		EF	9E	02002	MOVAB	P.AJW, R0	
		00000000G		00	16	02009	JSB	DBG\$CVT_MULH2_R1	
	51		C0	AD	9E	0200F	MOVAB	INTMED_DATA+16, R1	
	50	00000000'		EF	9E	02013	MOVAB	P.AJX, R0	
		00000000G		00	16	0201A	JSB	DBG\$CVT_MULH2_R1	
			30	AE	D7	02020	DECL	SCALE	
				D4	11	02023	BRB	418\$	
				27	18	02025	419\$:	BGEQ	420\$
	51		B0	AD	9E	02027	MOVAB	INTMED_DATA, R1	
	50	00000000'		EF	9E	0202B	MOVAB	P.AJY, R0	
		00000000G		00	16	02032	JSB	DBG\$CVT_DIVH2_R1	
	51		C0	AD	9E	02038	MOVAB	INTMED_DATA+16, R1	
	50	00000000'		EF	9E	0203C	MOVAB	P.AJZ, R0	
		00000000G		00	16	02043	JSB	DBG\$CVT_DIVH2_R1	
			30	AE	D6	02049	INCL	SCALE	
				D7	11	0204C	BRB	419\$	
01		0A		6B	8F	0204E	420\$:	CASEB	(R11), #10, #1
	0057			0047		02052	421\$:	.WORD	425\$-421\$,-
									427\$-421\$
01		0C		6B	8F	02056	422\$:	CASEB	(R11), #12, #1
	0026			000D		0205A	422\$:	.WORD	423\$-422\$,-
									424\$-422\$
		00000000'		EF	9F	0205E	PUSHAB	P.AKA	
				0F62	31	02064	BRW	647\$	
	50		B0	AD	9E	02067	423\$:	MOVAB	INTMED_DATA, R0
	51		34	AE	D0	0206B	MOVL	OUTPUT, R1	
		00000000G		00	16	0206F	JSB	DBG\$CVT_CVTHF_R1	
51	34		AE	04	C1	02075	ADDL3	#4, OUTPUT, RT	
	50		C0	AD	9E	0207A	MOVAB	INTMED_DATA+16, R0	
				21	11	0207E	BRB	426\$	
	50		B0	AD	9E	02080	424\$:	MOVAB	INTMED_DATA, R0
	51		34	AE	D0	02084	MOVL	OUTPUT, R1	
		00000000G		00	16	02088	JSB	DBG\$CVT_CVTHD_R1	
51	34		AE	08	C1	0208E	ADDL3	#8, OUTPUT, RT	
	50		C0	AD	9E	02093	MOVAB	INTMED_DATA+16, R0	
				18	11	02097	BRB	428\$	
	50		B0	AD	9E	02099	425\$:	MOVAB	INTMED_DATA, R0
	51		34	AE	D0	0209D	MOVL	OUTPUT, R1	
		00000000G		00	16	020A1	426\$:	JSB	DBG\$CVT_CVTHF_R1
				0E	11	020A7	BRB	429\$	
	50		B0	AD	9E	020A9	427\$:	MOVAB	INTMED_DATA, R0
	51		34	AE	D0	020AD	MOVL	OUTPUT, R1	
		00000000G		00	16	020B1	428\$:	JSB	DBG\$CVT_CVTHD_R1
				0F1E	31	020B7	429\$:	BRW	649\$
	02			6B	91	020BA	430\$:	CMPB	(R11), #2
				12	13	020BD	BEQL	433\$	
	0E			6B	91	020BF	CMPB	(R11), #14	
				0D	13	020C2	BEQL	433\$	
	25			6B	91	020C4	CMPB	(R11), #37	



			03	1E	020C7	BGEQU	432\$	
			0139	31	020C9	BRW	448\$	
		27	6B	91	020CC	CMPB	(R11), #39	
			F8	1A	020CF	BGTRU	431\$	
	58	AE	32	B0	020D1	MOVW	#50, CLASS_S_DESC	3370
	5C	AE	60	9E	020D5	MOVAB	TEMP BUF2, CLASS_S_DESC+4	3371
01		1B	04	BE	020DA	CASEB	@4(SP), #27, #1	3372
		004E	0004	8F	020DF	.WORD	435\$-434\$,-	
							438\$-434\$	
		57	0F	D0	020E3	MOVL	#15, DIGITS_IN_FRACT	3376
		52	03	D0	020E6	MOVL	#3, DIGITS_IN_EXP	3377
		50	07	D0	020E9	MOVL	#7, NOT_DIGITS	3378
50	F5	AD	10	00	ED	CMPZV	#0, #16, DST_INFO+5, NOT_DIGITS	3379
			15	15	020F?	BLEQ	437\$	
		51	F5	AD	3C	MOVZWL	DST_INFO+5, R1	3381
		51		50	C2	SUBL2	NOT_DIGITS, R1	
		50		57	D0	MOVL	DIGITS_IN_FRACT, R0	
		51		50	D1	CMPL	R0, R1	
			03	15	02101	BLEQ	436\$	
		50		51	D0	MOVL	R1, R0	
		57		50	D0	MOVL	R0, DIGITS_IN_FRACT	
				52	DD	PUSHL	DIGITS_IN_EXP	3382
				7E	D4	CLRL	-(SP)	
			38	AE	DD	PUSHL	SCALE	
				57	DD	PUSHL	DIGITS_IN_FRACT	
			68	AE	9F	PUSHAB	CLASS_S_DESC	
			80	AD	9F	PUSHAB	INTMED DATA	
	00000000G	00	06	FB	02118	CALLS	#6, FORSCVT_G_TE	
		6E	50	D0	0211F	MOVL	R0, STATUS	
		5F	6E	E8	02122	BLBS	STATUS, 442\$	3383
			00000000'	EF	9F	PUSHAB	P.AKB	
				48	11	BRB	441\$	
		57		21	D0	MOVL	#33, DIGITS_IN_FRACT	3388
		52		04	D0	MOVL	#4, DIGITS_IN_EXP	3389
		50		08	D0	MOVL	#8, NOT_DIGITS	3390
50	F5	AD	10	00	ED	CMPZV	#0, #16, DST_INFO+5, NOT_DIGITS	3391
				15	15	BLEQ	440\$	
		51	F5	AD	3C	MOVZWL	DST_INFO+5, R1	3393
		51		50	C2	SUBL2	NOT_DIGITS, R1	
		50		57	D0	MOVL	DIGITS_IN_FRACT, R0	
		51		50	D1	CMPL	R0, R1	
			03	15	0214B	BLEQ	439\$	
		50		51	D0	MOVL	R1, R0	
		57		50	D0	MOVL	R0, DIGITS_IN_FRACT	
				52	DD	PUSHL	DIGITS_IN_EXP	3394
				7E	D4	CLRL	-(SP)	
			38	AE	DD	PUSHL	SCALE	
				57	DD	PUSHL	DIGITS_IN_FRACT	
			68	AE	9F	PUSHAB	CLASS_S_DESC	
			80	AD	9F	PUSHAB	INTMED DATA	
	00000000G	00	06	FB	02162	CALLS	#6, FORSCVT_H_TE	
		6E	50	D0	02169	MOVL	R0, STATUS	
		15	6E	E8	0216C	BLBS	STATUS, 442\$	3395
			00000000'	EF	9F	PUSHAB	P.AKC	
				01	DD	PUSHL	#1	
			00028362	8F	DD	PUSHL	#164706	
	00000000G	00	03	FB	0217D	CALLS	#3, LIB\$SIGNAL	

60	AE	32	20	3B	02184	442\$:	SKPC	#32, #50, TEMP_BUF2	3399
			02	12	02189		BNEQ	443\$	
			51	D4	0218B		CLRL	R1	
	SA	50	60	AE	9E	0218D	443\$:	MOVAB	TEMP_BUF2, R0
	55	51	50	C3	02191		SUBL3	R0, R1, BUF_OFFSET	3400
		32	5A	C3	02195		SUBL3	BUF_OFFSET, #50, FINAL_LEN	3401
	14	AE	55	D0	02199		MOVL	FINAL_LEN, OUTPUT_STR_LEN	3405
		26	6B	91	0219D		CMPB	(R11), #38	
			2E	12	021A0		BNEQ	445\$	
	58	AE	55	B0	021A2		MOVW	FINAL_LEN, CLASS_S_DESC	3409
5C	AE	34	01	C1	021A6		ADDL3	#1, OUTPUT, CLASS_S_DESC+4	3410
		52	58	AE	9E	021AC	MOVAB	CLASS_S_DESC, R2	3411
		51	60	AE4A	9E	021B0	MOVAB	TEMP_BUF2[BUF_OFFSET], R1	
		50	55	D0	021B5		MOVL	FINAL_LEN, R0	
			00000000G	00	16	021B8	JSB	LIB\$COPY_R_DX6	
		6E	50	D0	021BE		MOVL	R0, STATUS	
	00000000G	8F	6E	D1	021C1		CMPL	STATUS, #LIB\$_STRTRU	3412
			03	13	021C8		BEQL	444\$	
			14DA	31	021CA		BRW	319\$	
			F4C2	31	021CD	444\$:	BRW	318\$	
		27	5B	91	021D0	445\$:	CMPB	(R11), #39	3417
			03	13	021D3		BEQL	446\$	
			F522	31	021D5		BRW	325\$	
	58	AE	55	B0	021D8	446\$:	MOVW	FINAL_LEN, CLASS_S_DESC	3421
5C	AE	34	AE	D0	021DC		MOVL	OUTPUT, CLASS_S_DESC+4	3422
	52	58	AE	9E	021E1		MOVAB	CLASS_S_DESC, R2	3423
	51	60	AE4A	9E	021E5		MOVAB	TEMP_BUF2[BUF_OFFSET], R1	
	50		55	D0	021EA		MOVL	FINAL_LEN, R0	
			00000000G	00	16	021ED	JSB	LIB\$COPY_R_DX6	
		6E	50	D0	021F3		MOVL	R0, STATUS	
	00000000G	8F	6E	D1	021F6		CMPL	STATUS, #LIB\$_STRTRU	3424
			03	13	021FD		BEQL	447\$	
			F4EF	31	021FF		BRW	323\$	
			F4D7	31	02202	447\$:	BRW	322\$	
			EF	9F	02205	448\$:	PUSHAB	P.AKD	3439
			0DBB	31	0220B		BRW	647\$	
		50	FD	3C	0220E	449\$:	MOVZWL	SRC_INFO+5, R0	3447
58	OA	A9	2C	50	D0	02212	MOVL	R0, NO_DIGITS	
		01		03	EF	02216	EXTZV	#3, #1, 10(R9), R8	3445
		6E		58	E9	0221C	BLBC	R0, 454\$	
		09	03	AA	91	0221F	CMPB	3(R10), #9	3447
				0A	12	02223	BNEQ	450\$	
		30	AE	08	AA	98	CVTBL	8(R10), SCALE	
		30	AE	30	AE	CE	MNEGL	SCALE, SCALE	
FF7C	CD	2C	AE	B0	AD	2C	AE	08 0222F 450\$:	
		58	AE	2C	AE	01	A1	02239	
		5C	AE	FF7C	CD	9E	0223F	ADDW3	#1, NO_DIGITS, CLASS_S_DESC
			7E	44	8F	9A	02245	MOVAB	TEMP_BUF1, CLASS_S_DESC+4
				34	AE	DD	02249	MOVZBL	#68, --(SP)
				7E	D4	0224C	PUSHL	SCALE	
				80	AD	9F	0224E	CLRL	-(SP)
				68	AE	9F	02251	PUSHAB	INTMED_DATA
					05	FB	02254	PUSHAB	CLASS_S_DESC
		00000000G	00	50	D0	0225B	CALLS	#5, OT\$CVT_T_H	
			6E	50	D0	0225B	MOVL	R0, STATUS	
			29	6E	E8	0225E	BLBS	STATUS, 453\$	
			50	00000000G	00	D0	02261	MOVL	DBG\$GL_OPCODE_NAME, R0

					30	AE	D5	02268	TSTL	SCALE	
						OC	18	0226B	BGEQ	451\$	
						50	DD	0226D	PUSHL	R0	
						01	DD	0226F	PUSHL	#1	
					0002869B	8F	DD	02271	PUSHL	#165531	
						0A	11	02277	BRB	452\$	
						50	DD	02279	PUSHL	R0	
						01	DD	0227B	PUSHL	#1	
					00028A02	8F	DD	0227D	PUSHL	#166402	
						03	FB	02283	CALLS	#3, LIB\$SIGNAL	
						00D1	31	0228A	BRW	462\$	
08	BE	01	B0	AD	2C	AE	37	0228D	CMPP4	NO_DIGITS, INTMED_DATA, #1, @PACK_ZERO	3471
						54	DC	02295	MOVPSL	R4	
		54		02		02	EF	02297	EXTZV	#2, #2, R4, R4	
						54	D7	0229C	DECL	R4	
						04	15	0229E	BLEQ	455\$	
			FF	AD		01	88	022A0	BISB2	#1, SRC_INFO+7	
					30	AE	D5	022A4	TSTL	SCALE	
						3C	13	022A7	BEQL	458\$	
						AE	34	022A9	MOVPS	NO_DIGITS, INTMED_DATA, TEMP_BUF1	
						OC	E9	022B1	BLBC	CVT_ROUND_FLAG, 456\$	
						AD	9E	022B5	MOVAB	INTMED_DATA, R5	
						54	AE	9E	022B9	MOVAB	NO_DIGITS, R4
			18	AE		05	D0	022BD	MOVL	#5, 24(SP)	
						0B	11	022C1	BRB	457\$	
						AD	9E	022C3	MOVAB	INTMED_DATA, R5	
						54	AE	9E	022C7	MOVAB	NO_DIGITS, R4
						18	AE	D4	022CB	CLRL	247(SP)
						53	AE	9E	022CE	MOVAB	24(SP), R3
						52	CD	9E	022D2	MOVAB	TEMP_BUF1, R2
						51	AE	9E	022D7	MOVAB	NO_DIGITS, R1
						50	AE	9E	022DB	MOVAB	SCALE, R0
					00000000G	00	16	022DF	JSB	DBG\$CVT_A\$HP_R1	
						50	6A	3C	022E5	MOVZWL	(R10), R0
						51	AA	98	022E8	CVTBL	8(R10), R1
						50	51	C0	022EC	ADDL2	R1, R0
						52	69	3C	022EF	MOVZWL	(R9), R2
						51	A9	98	022F2	CVTBL	8(R9), R1
						52	51	C0	022F6	ADDL2	R1, R2
						52	50	D1	022F9	CMPL	R0, R2
						58	15	022FC	BLEQ	461\$	
						15	BE	91	022FE	CMPB	@4(SP), #21
						52	12	02302	BNEQ	461\$	
						15	6B	91	02304	CMPB	(R11), #21
						4D	12	02307	BNEQ	461\$	
						50	AD	9E	02309	MOVAB	INTMED_DATA+15, HIGH_NIBBLE_PTR
						52	69	3C	0230D	MOVZWL	(R9), R2
						52	02	C6	02310	DIVL2	#2, R2
						52	50	C2	02313	SUBL2	HIGH_NIBBLE_PTR, R2
						52	52	CE	02316	MNEGL	R2, LOW_NIBBLE_PTR
						50	69	3C	02319	MOVZWL	(R9), R0
						50	01	7A	0231C	EMUL	#1, R0, #0, -(SP)
7E		00				02	7B	02321	EDIV	#2, (SP)+, R0, R0	
50		50				50	D5	02326	TSTL	R0	
						08	12	02328	BNEQ	459\$	
						00	EF	0232A	EXTZV	#0, #4, (LOW_NIBBLE_PTR), R0	
50		62			04	50	90	0232F	MOVAB	R0, (LOW_NIBBLE_PTR)	
					62						

[illegible]

3713

						F88C	31	023C2		BRW	383\$		
						B0	AD	B5	023C5	466\$:	TSTW	INTMED_DATA	3498
							03	12	023C8		BNEQ	467\$	
						053C	31	023CA		BRW	542\$		
	56	B1	AD			FF7C	CD	D4	023CD	467\$:	CLRL	TEMP_BUF1	3503
				B1	01		07	EF	023D1		EXTZV	#7, #1, INTMED_DATA+1, SIGN	3504
					AD		80	8F	8A	023D7	BICB2	#128, INTMED_DATA+1	3505
					54		B0	AD	3C	023DC	MOVZWL	INTMED_DATA, -FLOAT_SCALE	3506
					54		C000	C4	9E	023E0	MOVAB	-16384(R4), FLOAT_SCALE	
					52		08	A9	98	023E5	CVTBL	8(R9), R2	3507
					52			07	C0	023E9	ADDL2	#7, R2	
					52			54	D1	023EC	CMPL	FLOAT_SCALE, R2	
								5B	14	023EF	BGTR	474\$	
				FF7C	CD	40	8F	88	023F1		BISB2	#64, TEMP_BUF1	3512
					06		02	EF	023F7		EXTZV	#2, #6, INTMED_DATA+3, R0	3513
FF7C	50	B3	AD		00		50	F0	023FD		INSV	R0, #0, #6, TEMP_BUF1	
CD	CD		06		52		54	C3	02404		SUBL3	FLOAT_SCALE, R2, -FLOAT_SCALE	3514
							09	15	02408	468\$:	BLEQ	469\$	3515
				FF7C	CD		02	C6	0240A		DIVL2	#2, TEMP_BUF1	3517
							54	D7	0240F		DECL	FLOAT_SCALE	3518
							F5	11	02411		BRB	468\$	3515
					03		56	E8	02413	469\$:	BLBS	SIGN, 470\$	3520
						0545	31	02416		BRW	549\$		
					03	053B	31	02419	470\$:	BRW	548\$		
							58	E8	0241C	471\$:	BLBS	R8, 472\$	3532
						F852	31	0241F		BRW	385\$		
						B0	AD	B5	02422	472\$:	TSTW	INTMED_DATA	3545
							03	12	02425		BNEQ	473\$	
						0554	31	02427		BRW	553\$		
						FF7C	CD	D4	0242A	473\$:	CLRL	TEMP_BUF1	3550
	56	B1	AD		01		07	EF	0242E		EXTZV	#7, #1, INTMED_DATA+1, SIGN	3551
				B1	AD		80	8F	8A	02434	BICB2	#128, INTMED_DATA+1	3552
					54		B0	AD	3C	02439	MOVZWL	INTMED_DATA, -FLOAT_SCALE	3553
					54		C000	C4	9E	0243D	MOVAB	-16384(R4), FLOAT_SCALE	
					52		08	A9	98	02442	CVTBL	8(R9), R2	3554
					52			0F	C0	02446	ADDL2	#15, R2	
					52			54	D1	02449	CMPL	FLOAT_SCALE, R2	
							60	14	0244C	474\$:	BGTR	482\$	
				FF7D	CD	40	8F	88	0244E		BISB2	#64, TEMP_BUF1+1	3559
					0E		02	EF	02454		EXTZV	#2, #14, INTMED_DATA+2, R0	3560
					00		50	F0	0245A		INSV	R0, #0, #14, TEMP_BUF1	
FF7C	50	B2	AD		52		54	C3	02461		SUBL3	FLOAT_SCALE, R2, FLOAT_SCALE	3561
CD	CD		0E				09	15	02465	475\$:	BLEQ	476\$	3562
						FF7C	CD	02	C6	02467	DIVL2	#2, TEMP_BUF1	3564
							54	D7	0246C		DECL	FLOAT_SCALE	3565
							F5	11	0246E		BRB	475\$	3562
					03		56	E8	02470	476\$:	BLBS	SIGN, 477\$	3567
						055D	31	02473		BRW	560\$		
						0553	31	02476	477\$:	BRW	559\$		
					08		58	E8	02479	478\$:	BLBS	R8, 480\$	3579
					50	FF7C	CD	9E	0247C	479\$:	MOVAB	TEMP_BUF1, R0	3581
						0474	31	02481		BRW	540\$		
						B0	AD	B5	02484	480\$:	TSTW	INTMED_DATA	3587
							03	12	02487		BNEQ	481\$	
						047D	31	02489		BRW	542\$		
						FF7C	CD	D4	0248C	481\$:	CLRL	TEMP_BUF1	3592
	56	B1	AD		01		07	EF	02490		EXTZV	#7, #1, INTMED_DATA+1, SIGN	3593

			B1	AD	80	8F	8A	02496	BICB2	#128, INTMED_DATA+1	3594
				54	B0	AD	3C	0249B	MOVZWL	INTMED_DATA, -FLOAT_SCALE	3595
				54	C000	C4	9E	0249F	MOVAB	-16384(R4), FLOAT_SCALE	
				52	08	A9	98	024A4	CVTBL	8(R9), R2	3596
				52		07	C0	024AB	ADDL2	#7, R2	
				52		54	D1	024AB	CMPL	FLOAT_SCALE, R2	
						5A	14	024AE	BGTR	489\$	
			FF7C	CD	40	8F	88	024B0	BISB2	#64, TEMP_BUF1	3601
				06		02	EF	024B6	EXTZV	#2, #6, INTMED_DATA+3, R0	3602
FF7C	50	B3	AD	00		50	F0	024BC	INSV	R0, #0, #6, TEMP_BUF1	
	CD		06	54		54	C3	024C3	SUBL3	FLOAT_SCALE, R2, -FLOAT_SCALE	3603
				52		03	14	024C7	BGTR	484\$	3604
						FF47	31	024C9	BRW	469\$	
			FF7C	CD		02	C6	024CC	DIVL2	#2, TEMP_BUF1	3606
						54	D7	024D1	DECL	FLOAT_SCALE	3607
						F2	11	024D3	BRB	483\$	3604
				08		58	E8	024D5	BLBS	R8, 487\$	3621
				50	FF7C	CD	9E	024D8	MOVAB	TEMP_BUF1, R0	3623
						048D	31	024DD	BRW	551\$	
					B0	AD	B5	024E0	TSTW	INTMED_DATA	3629
						03	12	024E3	BNEQ	488\$	
						0496	31	024E5	BRW	553\$	
			FF7C	CD		D4	024E8	488\$:	CLRL	TEMP_BUF1	3634
	56	B1	AD	01		07	EF	024EC	EXTZV	#7, #1, INTMED_DATA+1, SIGN	3635
			B1	AD	80	8F	8A	024F2	BICB2	#128, INTMED_DATA+1	3636
				54	B0	AD	3C	024F7	MOVZWL	INTMED_DATA, -FLOAT_SCALE	3637
				54	C000	C4	9E	024FB	MOVAB	-16384(R4), FLOAT_SCALE	
				52	08	A9	98	02500	CVTBL	8(R9), R2	3638
				52		0F	C0	02504	ADDL2	#15, R2	
				52		54	D1	02507	CMPL	FLOAT_SCALE, R2	
						52	14	0250A	BGTR	494\$	
			FF7D	CD	40	8F	88	0250C	BISB2	#64, TEMP_BUF1+1	3643
				0E		02	EF	02512	EXTZV	#2, #14, INTMED_DATA+2, R0	3644
				00		50	F0	02518	INSV	R0, #0, #14, TEMP_BUF1	
FF7C	50	B2	AD	52		54	C3	0251F	SUBL3	FLOAT_SCALE, R2, FLOAT_SCALE	3645
	CD		0E			03	14	02523	BGTR	491\$	3646
						FF48	31	02525	BRW	476\$	
			FF7C	CD		02	C6	02528	DIVL2	#2, TEMP_BUF1	3648
						54	D7	0252D	DECL	FLOAT_SCALE	3649
						F2	11	0252F	BRB	490\$	3646
				63		58	E9	02531	BLBC	R8, 498\$	3663
					B0	AD	B5	02534	TSTW	INTMED_DATA	3671
						03	12	02537	BNEQ	493\$	
						04A8	31	02539	BRW	562\$	
			FF7C	CD		D4	0253C	493\$:	CLRL	TEMP_BUF1	3676
	56	B1	AD	01		07	EF	02540	EXTZV	#7, #1, INTMED_DATA+1, SIGN	3677
			B1	AD	80	8F	8A	02546	BICB2	#128, INTMED_DATA+1	3678
				54	B0	AD	3C	0254B	MOVZWL	INTMED_DATA, -FLOAT_SCALE	3679
				54	C000	C4	9E	0254F	MOVAB	-16384(R4), FLOAT_SCALE	
				52	08	A9	98	02554	CVTBL	8(R9), R2	3680
				52		1F	C0	02558	ADDL2	#31, R2	
				52		54	D1	0255B	CMPL	FLOAT_SCALE, R2	
						03	15	0255E	BLEQ	495\$	
						04AC	31	02560	BRW	565\$	
			FF7F	CD	40	8F	88	02563	BISB2	#64, TEMP_BUF1+3	3685
FF7D	CD		10	06	B2	AD	F0	02569	INSV	INTMED_DATA+2, #6, #16, TEMP_BUF1+1	3686
	50	B6	AD	0E		02	EF	02571	EXTZV	#2, #14, INTMED_DATA+6, R0	3687

FF7C	CD	OE	00	50	F0	02577	INSV	R0, #0, #14, TEMP_BUF1	:	
		54	52	54	C3	0257E	SUBL3	FLOAT_SCALE, R2, FLOAT_SCALE	:	3688
				09	15	02582	496\$: BLEQ	497\$	:	3689
	FF7C	CD		02	C6	02584	DIVL2	#2, TEMP_BUF1	:	3691
				54	D7	02589	DECL	FLOAT_SCALE	:	3692
				F5	11	0258B	BRB	496\$	:	3689
		07		56	E9	0258D	497\$: BLBC	SIGN, 498\$	:	3694
	FF7C	CD		CD	CE	02590	MNEGL	TEMP_BUF1, TEMP_BUF1	:	
	34	BE		FF7C	CD	D0	02597	498\$: MOVL	TEMP_BUF1, @OUTPUT	3695
				19	11	0259D	BRB	502\$	:	3476
		52		F5	AD	3C	0259F	499\$: MOVZWL	DST_INFO+5, R2	3706
		50		01	CE	025A3	MNEGL	#1, -1	:	3708
				0C	11	025A6	BRB	501\$	:	
	51	B0	AD	50	EF	025A8	500\$: EXTZV	I, #1, INTMED_DATA, R1	:	
34	BE		01	51	F0	025AE	INSV	R1, I, #1, @OUTPUT	:	
			F0	52	F2	025B4	501\$: AOBLS	R2, I, 500\$	:	3706
				0A1D	31	025B8	502\$: BRW	64\$	:	2187
				AD	3C	025BB	503\$: MOVZWL	SRC_INFO+5, NO DIGITS	:	3718
				2C	AE	37	025C0	CMPP4	NO_DIGITS, INTMED_DATA, #1, @PACK_ZERO	
				54	DC	025C8	MOVPSL	R4	:	
				02	EF	025CA	EXTZV	#2, #2, R4, R4	:	
				54	D7	025CF	DECL	R4	:	
				04	15	025D1	BLEQ	504\$	:	
				01	88	025D3	BISB2	#1, SRC_INFO+7	:	
				30	AE	D5	025D7	504\$: TSTL	SCALE	
				3C	13	025DA	BEQL	507\$	:	
				2C	AE	34	025DC	MOVZWL	NO DIGITS, INTMED_DATA, TEMP_BUF1	
				0C	AE	E9	025E4	BLBC	CVT_ROUND_FLAG, 505\$	
				55	B0	AD	9E	025E8	MOVAB	INTMED_DATA, R5
				54	2C	AE	9E	025EC	MOVAB	NO DIGITS, R4
				18	05	D0	025F0	MOVL	#5, 24(SP)	
				0B	11	025F4	BRB	506\$	:	
				55	B0	AD	9E	025F6	505\$: MOVAB	INTMED_DATA, R5
				54	2C	AE	9E	025FA	MOVAB	NO DIGITS, R4
				18	AE	D4	025FE	CLRL	24(SP)	
				53	18	AE	9E	02601	506\$: MOVAB	24(SP), R3
				52	FF7C	CD	9E	02605	MOVAB	TEMP_BUF1, R2
				51	2C	AE	9E	0260A	MOVAB	NO DIGITS, R1
				50	30	AE	9E	0260E	MOVAB	SCALE, R0
				00000000G	00	16	02612	JSB	DBG\$CVT_ASH_P_R1	
				50	6A	3C	02618	507\$: MOVZWL	(R10), R0	
				51	08	AA	98	0261B	CVTBL	8(R10), R1
				50	51	C0	0261F	ADDL2	R1, R0	
				52	69	3C	02622	MOVZWL	(R9), R2	
				51	08	A9	98	02625	CVTBL	8(R9), R1
				52	51	C0	02629	ADDL2	R1, R2	
				52	50	D1	0262C	CMPL	R0, R2	
				15	58	15	0262F	BLEQ	510\$	
				15	04	BE	91	02631	CMPB	@4(SP), #21
				52	12	02635	BNEQ	510\$	:	
				6B	91	02637	CMPB	(R11), #21	:	
				4D	12	0263A	BNEQ	510\$	:	
				50	BF	AD	9E	0263C	MOVAB	INTMED_DATA+15, HIGH_NIBBLE_PTR
				52	69	3C	02640	MOVZWL	(R9), R2	
				52	02	C6	02643	DIVL2	#2, R2	
				52	50	C2	02646	SUBL2	HIGH_NIBBLE_PTR, R2	
				52	52	CE	02649	MNEGL	R2, LOW_NIBBLE_PTR	

PC	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------



PC	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

PC	Op	OpC	OpD	OpI	OpR	OpS	OpT	OpU	OpV	OpW	OpX	OpY	OpZ	OpAA	OpAB	OpAC	OpAD	OpAE	OpAF	OpAG	OpAH	OpAI	OpAJ	OpAK	OpAL	OpAM	OpAN	OpAO	OpAP	OpAQ	OpAR	OpAS	OpAT	OpAU	OpAV	OpAW	OpAX	OpAY	OpAZ	OpBA	OpBB	OpBC	OpBD	OpBE	OpBF	OpBG	OpBH	OpBI	OpBJ	OpBK	OpBL	OpBM	OpBN	OpBO	OpBP	OpBQ	OpBR	OpBS	OpBT	OpBU	OpBV	OpBW	OpBX	OpBY	OpBZ	OpCA	OpCB	OpCC	OpCD	OpCE	OpCF	OpCG	OpCH	OpCI	OpCJ	OpCK	OpCL	OpCM	OpCN	OpCO	OpCP	OpCQ	OpCR	OpCS	OpCT	OpCU	OpCV	OpCW	OpCX	OpCY	OpCZ	OpDA	OpDB	OpDC	OpDD	OpDE	OpDF	OpDG	OpDH	OpDI	OpDJ	OpDK	OpDL	OpDM	OpDN	OpDO	OpDP	OpDQ	OpDR	OpDS	OpDT	OpDU	OpDV	OpDW	OpDX	OpDY	OpDZ	OpEA	OpEB	OpEC	OpED	OpEE	OpEF	OpEG	OpEH	OpEI	OpEJ	OpEK	OpEL	OpEM	OpEN	OpEO	OpEP	OpEQ	OpER	OpES	OpET	OpEU	OpEV	OpEW	OpEX	OpEY	OpEZ	OpFA	OpFB	OpFC	OpFD	OpFE	OpFF	OpFG	OpFH	OpFI	OpFJ	OpFK	OpFL	OpFM	OpFN	OpFO	OpFP	OpFQ	OpFR	OpFS	OpFT	OpFU	OpFV	OpFW	OpFX	OpFY	OpFZ	OpGA	OpGB	OpGC	OpGD	OpGE	OpGF	OpGG	OpGH	OpGI	OpGJ	OpGK	OpGL	OpGM	OpGN	OpGO	OpGP	OpGQ	OpGR	OpGS	OpGT	OpGU	OpGV	OpGW	OpGX	OpGY	OpGZ	OpHA	OpHB	OpHC	OpHD	OpHE	OpHF	OpHG	OpHH	OpHI	OpHJ	OpHK	OpHL	OpHM	OpHN	OpHO	OpHP	OpHQ	OpHR	OpHS	OpHT	OpHU	OpHV	OpHW	OpHX	OpHY	OpHZ	OpIA	OpIB	OpIC	OpID	OpIE	OpIF	OpIG	OpIH	OpII	OpIJ	OpIK	OpIL	OpIM	OpIN	OpIO	OpIP	OpIQ	OpIR	OpIS	OpIT	OpIU	OpIV	OpIW	OpIX	OpIY	OpIZ	OpJA	OpJB	OpJC	OpJD	OpJE	OpJF	OpJG	OpJH	OpJI	OpJJ	OpJK	OpJL	OpJM	OpJN	OpJO	OpJP	OpJQ	OpJR	OpJS	OpJT	OpJU	OpJV	OpJW	OpJX	OpJY	OpJZ	OpKA	OpKB	OpKC	OpKD	OpKE	OpKF	OpKG	OpKH	OpKI	OpKJ	OpKK	OpKL	OpKM	OpKN	OpKO	OpKP	OpKQ	OpKR	OpKS	OpKT	OpKU	OpKV	OpKW	OpKX	OpKY	OpKZ	OpLA	OpLB	OpLC	OpLD	OpLE	OpLF	OpLG	OpLH	OpLI	OpLJ	OpLK	OpLL	OpLM	OpLN	OpLO	OpLP	OpLQ	OpLR	OpLS	OpLT	OpLU	OpLV	OpLW	OpLX	OpLY	OpLZ	OpMA	OpMB	OpMC	OpMD	OpME	OpMF	OpMG	OpMH	OpMI	OpMJ	OpMK	OpML	OpMM	OpMN	OpMO	OpMP	OpMQ	OpMR	OpMS	OpMT	OpMU	OpMV	OpMW	OpMX	OpMY	OpMZ	OpNA	OpNB	OpNC	OpND	OpNE	OpNF	OpNG	OpNH	OpNI	OpNJ	OpNK	OpNL	OpNM	OpNN	OpNO	OpNP	OpNQ	OpNR	OpNS	OpNT	OpNU	OpNV	OpNW	OpNX	OpNY	OpNZ	OpOA	OpOB	OpOC	OpOD	OpOE	OpOF	OpOG	OpOH	OpOI	OpOJ	OpOK	OpOL	OpOM	OpON	OpOO	OpOP	OpOQ	OpOR	OpOS	OpOT	OpOU	OpOV	OpOW	OpOX	OpOY	OpOZ	OpPA	OpPB	OpPC	OpPD	OpPE	OpPF	OpPG	OpPH	OpPI	OpPJ	OpPK	OpPL	OpPM	OpPN	OpPO	OpPP	OpPQ	OpPR	OpPS	OpPT	OpPU	OpPV	OpPW	OpPX	OpPY	OpPZ	OpQA	OpQB	OpQC	OpQD	OpQE	OpQF	OpQG	OpQH	OpQI	OpQJ	OpQK	OpQL	OpQM	OpQN	OpQO	OpQP	OpQQ	OpQR	OpQS	OpQT	OpQU	OpQV	OpQW	OpQX	OpQY	OpQZ	OpRA	OpRB	OpRC	OpRD	OpRE	OpRF	OpRG	OpRH	OpRI	OpRJ	OpRK	OpRL	OpRM	OpRN	OpRO	OpRP	OpRQ	OpRR	OpRS	OpRT	OpRU	OpRV	OpRW	OpRX	OpRY	OpRZ	OpSA	OpSB	OpSC	OpSD	OpSE	OpSF	OpSG	OpSH	OpSI	OpSJ	OpSK	OpSL	OpSM	OpSN	OpSO	OpSP	OpSQ	OpSR	OpSS	OpST	OpSU	OpSV
----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

					00000000G	00	000286A3	01	DD	0288A	PUSHL	#1		
					34	BE		8F	DD	0288C	PUSHL	#165539		
							60	03	FB	02892	CALLS	#3, LIB\$SIGNAL		
								AE	B0	02899	532\$: MOVW	TEMP_BUF2, @OUTPUT		3850
							B0	6C	11	0289E	533\$: BRB	543\$		3844
								AD	B5	028A0	534\$: TSTW	INTMED_DATA		3857
								03	12	028A3	BNEQ	535\$		
								00D6	31	028A5	BRW	553\$		
							FF7C	CD	D4	028A8	535\$: CLRL	TEMP_BUF1		3862
56		B1	AD			01		07	EF	028AC	EXTZV	#7, #1, INTMED_DATA+1, SIGN		3863
					91	AD		8F	8A	028B2	BICB2	#128, INTMED_DATA+1		3864
						54		B0	3C	028B7	MOVZWL	INTMED_DATA, -FLOAT_SCALE		3865
						54	C000	C4	9E	028BB	MOVAB	-16384(R4), -FLOAT_SCALE		
						52	08	A9	98	028C0	CVTBL	8(R9), R2		3866
						52		0F	C0	028C4	ADDL2	#15, R2		
						52		54	D1	028C7	CMPL	FLOAT_SCALE, R2		
								64	14	028CA	536\$: BGTR	545\$		
					FF7D	CD	40	8F	88	028CC	BISB2	#64, TEMP_BUF1+1		3871
						0E		02	EF	028D2	EXTZV	#2, #14, INTMED_DATA+2, R0		3872
FF7C	50	B2	AD			00		50	F0	028D8	INSV	R0, #0, #14, TEMP_BUF1		
	CD		0E			52		54	C3	028DF	SUBL3	FLOAT_SCALE, R2, -FLOAT_SCALE		3873
			54					03	14	028E3	537\$: BGTR	538\$		3874
								FB88	31	028E5	BRW	476\$		
					FF7C	CD		02	C6	028E8	538\$: DIVL2	#2, TEMP_BUF1		3876
								54	D7	028ED	DECL	FLOAT_SCALE		3877
								F2	11	028EF	BRB	537\$		3874
						10		52	E8	028F1	539\$: BLBS	R2, 541\$		3891
						50	60	AE	9E	028F4	MOVAB	TEMP_BUF2, R0		3893
						51	34	AE	D0	028F8	540\$: MOVL	OUTPUT, R1		
							00000000G	00	16	028FC	JSB	DBG\$CVT_CVTLB_R1		
								7D	11	02902	BRB	554\$		
							B0	AD	B5	02904	541\$: TSTW	INTMED_DATA		3899
								05	12	02907	BNEQ	544\$		
							34	BE	94	02909	542\$: CLRB	@OUTPUT		3901
								73	11	0290C	543\$: BRB	554\$		
							FF7C	CD	D4	0290E	544\$: CLRL	TEMP_BUF1		3904
								07	EF	02912	EXTZV	#7, #1, INTMED_DATA+1, SIGN		3905
56		B1	AD			01		8F	8A	02918	BICB2	#128, INTMED_DATA+1		3906
					B1	AD		3C	0291D	MOVZWL	INTMED_DATA, -FLOAT_SCALE			3907
						54	C000	C4	9E	02921	MOVAB	-16384(R4), -FLOAT_SCALE		
						52	08	A9	98	02926	CVTBL	8(R9), R2		3908
						52		07	C0	0292A	ADDL2	#7, R2		
						52		54	D1	0292D	CMPL	FLOAT_SCALE, R2		
								73	14	02930	545\$: BGTR	556\$		
					FF7C	CD	40	8F	88	02932	BISB2	#64, TEMP_BUF1		3913
						06		02	EF	02938	EXTZV	#2, #6, INTMED_DATA+3, R0		3914
						00		50	F0	0293F	INSV	R0, #0, #6, TEMP_BUF1		
						52		54	C3	02945	SUBL3	FLOAT_SCALE, R2, -FLOAT_SCALE		3915
								09	15	02949	546\$: BLEQ	547\$		3916
					FF7C	CD		02	C6	0294B	DIVL2	#2, TEMP_BUF1		3918
								54	D7	02950	DECL	FLOAT_SCALE		3919
								F5	11	02952	BRB	546\$		3916
						07		56	E9	02954	547\$: BLBC	SIGN, 549\$		3921
					FF7C	CD	FF7C	CE	02957	548\$: MNEGL	TEMP_BUF1, TEMP_BUF1			
					34	BE	FF7C	CD	90	0295E	549\$: MOVB	TEMP_BUF1, @OUTPUT		3922
								1B	11	02964	BRB	554\$		3788
						10		52	E8	02966	550\$: BLBS	R2, 552\$		3933

5  
)

DBGCVTDX  
V04-000

M 14  
15-Sep-1984 23:57:30 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:16:44 [DEBUG.SRC]DBGCVTDX.B32;1

Page 158  
(29)

				50	60	AE	9E	02969		MOVAB	TEMP_BUF2, R0	3935	
				51	34	AE	D0	0296D	551\$:	MOVL	OUTPUT, R1		
					00000000G	00	16	02971		JSB	DBG\$CVT_CVTBW_R1		
						6E	11	02977		BRB	563\$		
					80	AD	B5	02979	552\$:	TSTW	INTMED_DATA	3941	
						05	12	0297C		BNEQ	555\$		
					34	BE	B4	0297E	553\$:	CLRW	@OUTPUT	3943	
						64	11	02981	554\$:	BRB	563\$		
					FF7C	CD	D4	02983	555\$:	CLRL	TEMP_BUF1	3946	
56		B1	AD		01	07	EF	02987		EXTZV	#7, #1, INTMED_DATA+1, SIGN	3947	
				B1	AD	8F	8A	0298D		BICB2	#128, INTMED_DATA+1	3948	
					54	AD	3C	02992		MOVZWL	INTMED_DATA, -FLOAT_SCALE	3949	
					54	C000	C4	9E	02996	MOVAB	-16384(R4), FLOAT_SCALE		
					52	08	A9	98	0299B	CVTBL	8(R9), R2	3950	
					52		0F	C0	0299F	ADDL2	#15, R2		
					52		54	D1	029A2	CMPL	FLOAT_SCALE, R2		
							68	14	029A5	556\$:	BGTR	565\$	
							8F	88	029A7	BISB2	#64, TEMP_BUF1+1	3955	
				FF7D	CD		02	EF	029AD	EXTZV	#2, #14, INTMED_DATA+2, R0	3956	
FF7C	50		B2	AD	0E		50	F0	029B3	INSV	R0, #0, #14, TEMP_BUF1		
CD	CD			OE	00		54	C3	029BA	SUBL3	FLOAT_SCALE, R2, FLOAT_SCALE	3957	
				54	52		09	15	029BE	557\$:	BLEQ	558\$	3958
							02	C6	029C0	DIVL2	#2, TEMP_BUF1	3960	
				FF7C	CD		54	D7	029C5	DECL	FLOAT_SCALE	3961	
							F5	11	029C7	BRB	557\$	3958	
					07		56	E9	029C9	558\$:	BLBC	SIGN, 560\$	3963
				FF7C	CD	FF7C	CD	CE	029CC	559\$:	MNEGL	TEMP_BUF1, TEMP_BUF1	
				34	BE	FF7C	CD	B0	029D3	560\$:	MOVW	TEMP_BUF1, @OUTPUT	3964
							79	11	029D9	BRB	570\$	3788	
					71		52	E9	029DB	561\$:	BLBC	R2, 569\$	3975
						FF7C	CD	B5	029DE		TSTW	TEMP_BUF1	3983
							05	12	029E2	BNEQ	564\$		
					34	BE	D4	029E4	562\$:	CLRL	@OUTPUT	3985	
							6B	11	029E7	563\$:	BRB	570\$	
					60	AE	D4	029E9	564\$:	CLRL	TEMP_BUF2	3988	
							07	EF	029EC	EXTZV	#7, #1, TEMP_BUF1+1, SIGN	3989	
							8F	8A	029F3	BICB2	#128, TEMP_BUF1+1	3990	
				FF7D	CD	FF7C	CD	3C	029F9	MOVZWL	TEMP_BUF1, -FLOAT_SCALE	3991	
					54	C000	C4	9E	029FE	MOVAB	-16384(R4), FLOAT_SCALE		
					52	08	A9	98	02A03	CVTBL	8(R9), R2	3992	
					52		1F	C0	02A07	ADDL2	#31, R2		
					52		54	D1	02A0A	CMPL	FLOAT_SCALE, R2		
							11	15	02A0D	BLEQ	566\$		
						00000000G	00	DD	02A0F	565\$:	PUSHL	DBG\$GL_OPCODE_NAME	3994
							01	DD	02A15		PUSHL	#1	
						000286A3	8F	DD	02A17		PUSHL	#165539	
							05B1	31	02A1D	BRW	648\$		
							8F	88	02A20	566\$:	BISB2	#64, TEMP_BUF2+3	3997
61	AE			63	AE		CD	F0	02A25	INSV	TEMP_BUF1+2, #6, #16, TEMP_BUF2+1	3998	
	50				06	FF7E	02	EF	02A2D	EXTZV	#2, #14, TEMP_BUF1+6, R0	3999	
60	AE		82	AD	0E		50	F0	02A33	INSV	R0, #0, #14, TEMP_BUF2		
				OE	00		54	C3	02A39	SUBL3	FLOAT_SCALE, R2, FLOAT_SCALE	4000	
				54	52		08	15	02A3D	567\$:	BLEQ	568\$	4001
							02	C6	02A3F	DIVL2	#2, TEMP_BUF2	4003	
				60	AE		54	D7	02A43	DECL	FLOAT_SCALE	4004	
							F6	11	02A45	BRB	567\$	4001	
					05		56	E9	02A47	568\$:	BLBC	SIGN, 569\$	4006

9

9

5

4

5

0

1

3

6

7

4

50

50

02	50	DC	02B0B	MOVPSL	R0		
15	02	EF	02B0D	EXTZV	#2, #2, R0, R0		
	50	F4	02B12	SOBGEQ	R0, 581\$		
	00	DD	02B15	PUSHL	DBG\$GL_OPCODE_NAME		
	01	DD	02B18	PUSHL	#1		
	000286A3	8F	DD	PUSHL	#165539		
00000000G	00	03	FB	CALLS	#3, LIB\$SIGNAL		
	00E0	8F	B9	BICPSW	#224		4042
	09	OC	AE	BLBC	CVT_ROUND_FLAG, 582\$		4043
34	BE	FF7C	CD	CVTRDL	TEMP_BUF1, @OUTPUT		4045
			31	BRW	583\$		
34	BE	FF7C	CD	CVTDL	TEMP_BUF1, @OUTPUT		4047
	00E0	8F	B8	BISPSW	#224		4049
		4C	11	BRB	587\$		4030
58	AE	FD	AD	MOVW	SRC_INFO+5, CLASS_S_DESC		4054
5C	AE	F9	AD	MOVL	SRC_INFO+1, CLASS_S_DESC+4		4055
	7E	55	8F	MOVZBL	#85, -(SP)		4057
	7E	34	AE	MNEGL	SCALE, -(SP)		4056
			7E	CLRL	-(SP)		
		FF7C	CD	PUSHAB	TEMP_BUF1		
		68	AE	PUSHAB	CLASS_S_DESC		
00000000G	00	05	FB	CALLS	#5, OTS\$CVT_T_H		
	6E	50	DD	MOVL	R0, STATUS		
	15	6E	EB	BLBS	STATUS, 585\$		4058
		00000000G	00	PUSHL	DBG\$GL_OPCODE_NAME		
			01	PUSHL	#1		
		00028298	8F	PUSHL	#164504		
00000000G	00	03	FB	CALLS	#3, LIB\$SIGNAL		
	50	FF7C	CD	MOVAB	TEMP_BUF1, R0		4059
	51	34	AE	MOVL	OUTPUT, R1		
		00000000G	00	JSB	DBG\$CVT_CVTRHQ_R1		
			04	BRW	649\$		
			31	CLRL	SIGN_FLAG		4063
	52	34	AE	MOVL	OUTPUT, R2		4073
		08	A2	CLRL	8(R2)		4074
		04	A2	CLRL	4(R2)		4075
	62	F9	BD	MOVZBL	@SRC_INFO+1, (R2)		4076
	62		30	SUBL2	#48, (R2)		
			02	BGEQ	589\$		4081
			05	BRB	590\$		
	09		62	CMPL	(R2), #9		
			2F	BLEQ	592\$		
FFFFFFFFD	8F		62	CMPL	(R2), #-3		4083
			12	BNEQ	591\$		
		F9	AD	INCL	SRC_INFO+1		4086
		FD	AD	DECW	SRC_INFO+5		4087
	62	F9	BD	MOVZBL	@SRC_INFO+1, (R2)		4088
	62		30	SUBL2	#48, (R2)		
	58		01	MOVL	#1, SIGN_FLAG		4090
			14	BRB	592\$		4083
		F9	AD	PUSHL	SRC_INFO+1		4093
			01	PUSHL	#1		
			02	PUSHL	#2		
		00028AAA	8F	PUSHL	#166570		
00000000G	00	04	FB	CALLS	#4, LIB\$SIGNAL		
	56	FD	AD	MOVZWL	SRC_INFO+5, R6		4100
			53	CLRL	CURRENT_CHAR_NUM		4103

PC	Instruction	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418
----	-------------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

			49	11	02CB6	603\$:	BRB	606\$	4147
	7E	55	8F	9A	02CB8	604\$:	MOVZBL	#85, -(SP)	4172
	7E	34	AE	CE	02CBC		MNEGL	SCALE, -(SP)	4171
			7E	D4	02CC0		CLRL	-(SP)	
		FF7C	CD	9F	02CC2		PUSHAB	TEMP_BUF1	
		68	AE	9F	02CC6		PUSHAB	CLASS_S_DESC	
00C00000G	00		05	FB	02CC9		CALLS	#5, OTSSCVT-T_H	
	6E		50	D0	02CD0		MOVL	R0, STATUS	
	15		6E	E8	02CD3		BLBS	STATUS, 605\$	4173
		00000000G	00	DD	02CD6		PUSHL	DBG\$GL_OPCODE_NAME	
			01	DD	02CDC		PUSHL	#1	
		00028298	8F	DD	02CDE		PUSHL	#164504	
00000000G	00		03	FB	02CE4		CALLS	#3, LIB\$SIGNAL	
	58	34	AE	D0	02CEB	605\$:	MOVL	OUTPUT, R8	4174
68	FF7C		10	28	02CEF		MOVCS	#16, TEMP_BUF1, (R8)	
			6B	91	02CF5		CMPB	(R11), #30	4175
			07	12	02CF8		BNEQ	606\$	
10	00		00	2C	02CFA		MOVCS	#0, (SP), #0, #16, 16(R8)	4181
			AB		02CFF				
			02D4	31	02D01	606\$:	BRW	649\$	2187
	02		6B	91	02D04	607\$:	CMPB	(R11), #2	4193
			12	13	02D07		BEQL	610\$	
	0E		6B	91	02D09		CMPB	(R11), #14	
			0D	13	02D0C		BEQL	610\$	
	25		6B	91	02D0E		CMPB	(R11), #37	
			03	1E	02D11		BGEQU	609\$	
			0252	31	02D13	608\$:	BRW	643\$	
	27		6B	91	02D16	609\$:	CMPB	(R11), #39	
			F8	1A	02D19		BGTRU	608\$	
		30	AE	D5	02D1B	610\$:	TSTL	SCALE	4195
			03	12	02D1E		BNEQ	611\$	
			015C	31	02D20		BRW	630\$	
	58	AE	FD	AD	02D23	611\$:	MOVW	SRC_INFO+5, CLASS_S_DESC	4198
5C	AE	F9	AD	D0	02D28		MOVL	SRC_INFO+1, CLASS_S_DESC+4	4199
	7E	55	8F	9A	02D2D		MOVZBL	#85, -(SP)	4201
	7E	34	AE	CE	02D31		MNEGL	SCALE, -(SP)	4200
			7E	D4	02D35		CLRL	-(SP)	
		FF7C	CD	9F	02D37		PUSHAB	TEMP_BUF1	
		68	AE	9F	02D3B		PUSHAB	CLASS_S_DESC	
00000000G	00		05	FB	02D3E		CALLS	#5, OTSSCVT-T_H	
	6E		50	D0	02D45		MOVL	R0, STATUS	
	03		6E	E8	02D48		BLBS	STATUS, 612\$	4202
			0120	31	02D4B		BRW	629\$	
	58	AE	32	B0	02D4E	612\$:	MOVW	#50, CLASS_S_DESC	4205
5C	AE	60	AE	9E	02D52		MOVAB	TEMP_BUF2, CLASS_S_DESC+4	4206
	09	F5	AD	B1	02D57		CMPW	DST_INFO+5, #9	4207
			05	1A	02D5B		BGTRU	613\$	
	57		21	D0	02D5D		MOVL	#33, DIGITS_IN_FRACT	4209
			12	11	02D60		BRB	615\$	
	50	F5	AD	3C	02D62	613\$:	MOVZWL	DST_INFO+5, R0	4211
	50		09	C2	02D66		SUBL2	#9, R0	
	21		50	D1	02D69		CMPB	R0, #33	
			03	15	02D6C		BLEQ	614\$	
	50		21	D0	02D6E		MOVL	#33, R0	
	57		50	D0	02D71	614\$:	MOVL	R0, DIGITS_IN_FRACT	
			04	DD	02D74	615\$:	PUSHL	#4	4212
			7E	7C	02D76		CLRL	-(SP)	



			68	57	DD	02D78	PUSHL	DIGITS IN FRACT		
			FF7C	AE	9F	02D7A	PUSHAB	CLASS S DESC		
				CD	9F	02D7D	PUSHAB	TEMP_BUF1		
	00000000G	00		06	FB	02D81	CALLS	#6, FOR\$CVT_H_TE		
		6E		50	D0	02D88	MOVL	R0, STATUS		
		02		6E	E9	02D8B	BLBC	STATUS, 616\$	4213	
				15	11	02D8E	BRB	617\$		
			00000000'	EF	9F	02D90	PUSHAB	P.AKJ		
				01	DD	02D96	PUSHL	#1		
			00028362	8F	DD	02D98	PUSHL	#164706		
	00000000G	00		03	FB	02D9E	CALLS	#3, LIB\$SIGNAL		
60	AE	32		20	3B	02DA5	SKPC	#32, #50, TEMP_BUF2	4214	
				02	12	02DAA	BNEQ	618\$		
				51	D4	02DAC	CLRL	R1		
		50	60	AE	9E	02DAE	MOVAB	TEMP_BUF2, R0		
	SA	51		50	C3	02DB2	SUBL3	R0, R1, BUF_OFFSET		
	SS	32		5A	C3	02DB6	SUBL3	BUF_OFFSET, #50, FINAL_LEN	4215	
		14	AE	55	D0	02DBA	MOVL	FINAL_LEN, OUTPUT_STR_LEN	4216	
			26	6B	91	02DBE	CMPB	(R11), #38	4220	
				51	12	02DC1	BNEQ	623\$		
	58	AE		55	B0	02DC3	MOVW	FINAL_LEN, CLASS S DESC	4224	
5C	AE	34	AE	01	C1	02DC7	ADDL3	#1, OUTPUT, CLASS_S_DESC+4	4225	
			52	AE	9E	02DCD	MOVAB	CLASS S DESC, R2	4226	
			51	60	AE	4A	9E	02DD1	MOVAB	TEMP_BUF2[BUF_OFFSET], R1
			50	55	D0	02DD6	MOVL	FINAL_LEN, R0		
			00000000G	00	16	02DD9	JSB	LIB\$COPY_R_DX6		
	00000000G	6E		50	D0	02DDF	MOVL	R0, STATUS		
		8F		6E	D1	02DE2	CMPB	STATUS, #LIB\$_STRTRU	4227	
				02	13	02DE9	BEQL	619\$		
				15	11	02DEB	BRB	620\$		
			00000000G	00	DD	02DED	PUSHL	DBG\$GL_OPCODE_NAME		
				01	DD	02DF3	PUSHL	#1		
			000286AB	8F	DD	02DF5	PUSHL	#165547		
	00000000G	00		03	FB	02DFB	CALLS	#3, LIB\$SIGNAL		
		09		6E	E8	02E02	BLBS	STATUS, 622\$	4228	
				6E	DD	02E05	PUSHL	STATUS		
	00000000G	00		01	FB	02E07	CALLS	#1, LIB\$SIGNAL		
		34	BE	55	90	02E0E	MOVW	FINAL_LEN, @OUTPUT	4229	
				58	11	02E12	BRB	628\$	4218	
			27	6B	91	02E14	CMPB	(R11), #39	4232	
				03	13	02E17	BEQL	624\$		
				31	02E19	BRW	325\$			
		58	AE	55	B0	02E1C	MOVW	FINAL_LEN, CLASS S DESC	4236	
	5C	AE	34	AE	D0	02E20	MOVL	OUTPUT, CLASS_S_DESC+4	4237	
		52	58	AE	9E	02E25	MOVAB	CLASS S DESC, R2	4238	
		51	60	AE	4A	9E	02E29	MOVAB	TEMP_BUF2[BUF_OFFSET], R1	
		50		55	D0	02E2E	MOVL	FINAL_LEN, R0		
			00000000G	00	16	02E31	JSB	LIB\$COPY_R_DX6		
	00000000G	6E		50	D0	02E37	MOVL	R0, STATUS		
		8F		6E	D1	02E3A	CMPB	STATUS, #LIB\$_STRTRU	4239	
				15	12	02E41	BNEQ	625\$		
			00000000G	00	DD	02E43	PUSHL	DBG\$GL_OPCODE_NAME		
				01	DD	02E49	PUSHL	#1		
			000286AB	8F	DD	02E4B	PUSHL	#165547		
	00000000G	00		03	FB	02E51	CALLS	#3, LIB\$SIGNAL		
		09		6E	E8	02E58	BLBS	STATUS, 627\$	4240	
				6E	DD	02E5B	PUSHL	STATUS		

50	00000000G	00	01	FB	02E5D	CALLS	#1, LIB\$SIGNAL	
		55	34	AE	C1 02E64	627\$:	ADDL3	OUTPUT, FINAL_LEN, R0
			01	AO	94 02E69		CLRB	1(R0)
			64	11	02E6C	628\$:	BRB	634\$
	00000000G	00	DD	02E6E	629\$:	PUSHL	DBG\$GL_OPCODE_NAME	
		01	DD	02E74		PUSHL	#1	
	00028298	8F	DD	02E76		PUSHL	#164504	
		0152	31	02E7C		BRW	648\$	
	14	AE	6A	3C 02E7F	630\$:	MOVZWL	(R10), OUTPUT_STR_LEN	4257
		26	6B	91 02E83		CMPB	(R11), #38	4260
			4D	12 02E86		BNEQ	635\$	
	58	AE	6A	B0 02E88		MOVW	(R10), CLASS_S_DESC	4264
5C	AE	34	AE	01 C1 02E8C		ADDL3	#1, OUTPUT, CLASS_S_DESC+4	4265
		51	AE	9E 02E92		MOVAB	CLASS_S_DESC, R1	4266
		50	5A	D0 02E96		MOVL	R10, R0	
	00000000G	00	16	02E99		JSB	LIB\$SCOPY DXDX6	
		6E	50	D0 02E9F		MOVL	R0, STATUS	
	00000000G	8F	6E	D1 02EA2		CMPL	STATUS, #LIB\$_STRTRU	4267
			02	13 02EA9		BEQL	631\$	
			15	11 02EAB		BRB	632\$	
	00000000G	00	DD	02EAD	631\$:	PUSHL	DBG\$GL_OPCODE_NAME	
		01	DD	02EB3		PUSHL	#1	
	000286AB	8F	DD	02EB5		PUSHL	#165547	
	00000000G	00	03	FB 02EBB		CALLS	#3, LIB\$SIGNAL	
		09	6E	E8 02EC2	632\$:	BLBS	STATUS, 633\$	4268
			6E	DD 02EC5		PUSHL	STATUS	
	00000000G	00	01	FB 02EC7		CALLS	#1, LIB\$SIGNAL	
		34	BE	6A 90 02ECE	633\$:	MOVW	(R10), @OUTPUT	4269
			0091	31 02ED2	634\$:	BRW	642\$	4258
		27	6B	91 02ED5	635\$:	CMPB	(R11), #39	4272
			51	12 02ED8		BNEQ	639\$	
	58	AE	6A	B0 02EDA		MOVW	(R10), CLASS_S_DESC	4276
	5C	AE	AE	D0 02EDE		MOVL	OUTPUT, CLASS_S_DESC+4	4277
		51	AE	9E 02EE3		MOVAB	CLASS_S_DESC, RT	4278
		50	5A	D0 02EE7		MOVL	R10, R0	
	00000000G	00	16	02EEA		JSB	LIB\$SCOPY DXDX6	
		6E	50	D0 02EF0		MOVL	R0, STATUS	
	00000000G	8F	6E	D1 02EF3		CMPL	STATUS, #LIB\$_STRTRU	4279
			02	13 02EFA		BEQL	636\$	
			15	11 02EFC		BRB	637\$	
	00000000G	00	DD	02EFE	636\$:	PUSHL	DBG\$GL_OPCODE_NAME	
		01	DD	02F04		PUSHL	#1	
	000286AB	8F	DD	02F06		PUSHL	#165547	
	00000000G	00	03	FB 02F0C		CALLS	#3, LIB\$SIGNAL	
		09	6E	E8 02F13	637\$:	BLBS	STATUS, 638\$	4280
			6E	DD 02F16		PUSHL	STATUS	
	00000000G	00	01	FB 02F18		CALLS	#1, LIB\$SIGNAL	
		50	6A	3C 02F1F	638\$:	MOVZWL	(R10), R0	4281
		50	AE	C0 02F22		ADDL2	OUTPUT, R0	
			60	94 02F26		CLRB	(R0)	
		0096	31	02F28		BRW	645\$	4258
		51	59	D0 02F2B	639\$:	MOVL	R9, R1	4286
		50	5A	D0 02F2E		MOVL	R10, R0	
	00000000G	00	16	02F31		JSB	LIB\$SCOPY DXDX6	
		6E	50	D0 02F37		MOVL	R0, STATUS	
	00000000G	8F	6E	D1 02F3A		CMPL	STATUS, #LIB\$_STRTRU	4287
			02	13 02F41		BEQL	640\$	

[illegible]

```
; Routine Size: 12460 bytes,   Routine Base: DBG$CODE + 02F0
```

```

4256 4360 1 ROUTINE CVT_HANDLER (SIG, MECH) =
4257 4361 1
4258 4362 1 FUNCTIONAL DESCRIPTION:
4259 4363 1
4260 4364 1 This handler will resignal opcode reserved to digital; it
4261 4365 1 otherwise translates system service signals to debug
4262 4366 1 error codes and resignals.
4263 4367 1
4264 4368 1 FORMAL PARAMETERS:
4265 4369 1
4266 4370 1 SIG rr.r A counted vector of parameters describing the condition.
4267 4371 1 MECH rr.r A counted vector of parameters from CHF.
4268 4372 1
4269 4373 1 IMPLICIT INPUTS:
4270 4374 1
4271 4375 1 NONE
4272 4376 1
4273 4377 1 IMPLICIT OUTPUTS:
4274 4378 1
4275 4379 1 NONE
4276 4380 1
4277 4381 1 COMPLETION STATUS: (or ROUTINE VALUE:)
4278 4382 1
4279 4383 1 $$$ RESIGNAL when opcode reserved to digital exception. Any other case
4280 4384 1 will result in a debug condition being signalled.
4281 4385 1
4282 4386 1 SIDE EFFECTS:
4283 4387 1
4284 4388 1 NONE
4285 4389 1
4286 4390 2 BEGIN
4287 4391 2 MAP
4288 4392 2 SIG : REF VECTOR,
4289 4393 2 MECH : REF VECTOR;
4290 4394 2
4291 4395 2
4292 4396 2 ! Translate error code if this is not an UNWIND, or opcode reserved to digital.
4293 4397 2 ! Otherwise, signal debug error.
4294 4398 2
4295 4399 2 IF (LIB$MATCH_COND (SIG [1], %REF ($$$_UNWIND), %REF ($$$_OPCODE))) GTR 0
4296 4400 2 THEN
4297 4401 2 RETURN ($$$_RESIGNAL);
4298 4402 2
4299 4403 2
4300 4404 2 ! Translate all numeric exceptions to debug's facility code.
4301 4405 2 ! Also, translate $$$_ROPRAND to $$$_ROPRANDF.
4302 4406 2
4303 4407 2 ! SELECTONE .SIG[1] OF
4304 4408 2 SET
4305 4409 2 [$$$_INTOVF]:
4306 4410 2 SIGNAL (DBG$_INTOVF, 1, .DBG$_GL_OPCODE_NAME);
4307 4411 2 [$$$_DECOVF]:
4308 4412 2 SIGNAL (DBG$_DECOVF, 1, .DBG$_GL_OPCODE_NAME);
4309 4413 2 [$$$_FLTTOVF, $$$_FLTTOVF_F]:
4310 4414 2 SIGNAL (DBG$_FLTTOVF, 1, .DBG$_GL_OPCODE_NAME);
4311 4415 2 [$$$_FLTUND, $$$_FLTUND_F]:
4312 4416 2 BEGIN

```

```

        .SAVE RESULT = 0;
        SIGNAL (DBG$_IFLUND, 1, .DBG$GL_OPCODE_NAME);
    END;
[SS$ROPRAND]:
    BEGIN
        IF .DECIMAL_CONVERT
        THEN
            SIGNAL (DBG$_DECROPRAND)
        ELSE
            SIGNAL (DBG$_ROPRANDF, 1, .DBG$GL_OPCODE_NAME);
        END;
    [OTHERWISE]:
        RETURN (SS$_RESIGNAL);
    TES;

SETUNWIND();
RETURN 0;
END;
                                ! End of CVT HA

```

```
! End of CVT_HANDLER
```

Offset	Instruction	Comment	Address
00000000	00 9E 00002	WORD	4360
00000000	00 9E 00009	LIB\$SIGNAL, R4	
00000000	08 C2 00010	DBG\$GL_OPCODE_NAME, R3	
04 AE 043C	8F 3C 00013	#8, SP	4399
04 AE 0920	8F 3C 0001C	#1084, 4(SP)	
52 04	AE 9F 00019	4(SP)	
04 AE 0920	8F 3C 0001C	#2336, 4(SP)	
52 04	AE 9F 00022	4(SP)	
00000000G 00 04	AC D0 00025	MOVL SIG, R2	
00000000G 00 04	A2 9F 00029	PUSHAB 4(R2)	
00000000G 00 03	FB 0002C	CALLS #3, LIB\$MATCH_COND	
00000000G 00 50	D5 00033	TSTL R0	
00000000G 00 77	14 00035	BGTR 7\$	
0000047C 52 04	A2 D0 00037	MOVL 4(R2), R2	4407
0000047C 8F	52 D1 0003B	CMPL R2, #1148	4409
0000047C 8F	0C 12 00042	BNEQ 1\$	
0000047C 8F	63 DD 00044	PUSHL DBG\$GL_OPCODE_NAME	4410
0000047C 8F	01 DD 00046	PUSHL #1	
0000047C 8F	8F DD 00048	PUSHL #165539	
000004A4 8F	7C 11 0004E	BRB 9\$	
000004A4 8F	52 D1 00050	CMPL R2, #1188	4411
000004A4 8F	0C 12 00057	BNEQ 2\$	
000004A4 8F	63 DD 00059	PUSHL DBG\$GL_OPCODE_NAME	4412
000004A4 8F	01 DD 0005B	PUSHL #1	
000004A4 8F	8F DD 0005D	PUSHL #166458	
0000048C 8F	67 11 00063	BRB 9\$	
0000048C 8F	52 D1 00065	CMPL R2, #1164	4413
0000048C 8F	09 13 0006C	BEQL 3\$	
000004B4 8F	52 D1 0006E	CMPL R2, #1204	
000004B4 8F	0C 12 00075	BNEQ 4\$	
000004B4 8F	63 DD 00077	PUSHL DBG\$GL_OPCODE_NAME	4414
000004B4 8F	01 DD 00079	PUSHL #1	
000004B4 8F	8F DD 0007B	PUSHL #166402	

DBGCVTDX  
V04-000

K 15  
15-Sep-1984 23:57:30 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:16:44 [DEBUG.SRC]DBGCVTDX.B32;1

Page 169  
(30)

0000049C	8F	49	11	00081	BRB	9\$	:	
		52	D1	00083	4\$: CMPL	R2, #1180	:	4415
000004C4	8F	09	13	0008A	BEQL	5\$	:	
		52	D1	0008C	CMPL	R2, #1220	:	
		12	12	00093	BNEQ	6\$	:	
	00000000'	FF	D4	00095	5\$: CLRL	@SAVE_RESULT	:	4417
		63	DD	0009B	PUSHL	DBG\$GC_OPCODE_NAME	:	4418
		01	DD	0009D	PUSHL	#1	:	
	0002869B	8F	DD	0009F	PUSHL	#165531	:	
		25	11	000A5	BRB	9\$	:	
00000454	8F	52	D1	000A7	6\$: CMPL	R2, #1108	:	4420
		21	12	000AE	7\$: BNEQ	10\$	:	
	0B 00000000'	EF	E9	000B0	BLBC	DECIMAL_CONVERT, 8\$	:	4422
	00028A42	8F	DD	000B7	PUSHL	#166466	:	4424
	64	01	FB	000BD	CALLS	#1, LIB\$SIGNAL	:	
		15	11	000C0	BRB	11\$	:	
		63	DD	000C2	8\$: PUSHL	DBG\$GL_OPCODE_NAME	:	4426
		01	DD	000C4	PUSHL	#1	:	
	00028A0A	8F	DD	000C6	PUSHL	#166410	:	
	64	03	FB	000CC	9\$: CALLS	#3, LIB\$SIGNAL	:	
		06	11	000CF	BRB	11\$	:	4407
	50 0918	8F	3C	000D1	10\$: MOVZWL	#2328, R0	:	4429
		04	000D6	RET			:	
00000000G	00	7E	7C	000D7	11\$: CLRQ	-(SP)	:	4432
		02	FB	000D9	CALLS	#2, SYSSUNWIND	:	
		50	D4	000E0	CLRL	R0	:	4433
		04	000E2	RET			:	4434

; Routine Size: 227 bytes, Routine Base: DBG\$CODE + 339C

```

: 4332 4435 1 ROUTINE FIND_CVT_PATH (SOURCE, DESTINATION, SRC_INFO, DST_INFO, CVT_PATH) =
: 4333 4436 1
: 4334 4437 1 FUNCTION
: 4335 4438 1 This routine is called by DBG$CVT_DX_DX, and has the following four
: 4336 4439 1 functions:
: 4337 4440 1
: 4338 4441 1 a. It finds any errors concerning the class and data type of the source
: 4339 4442 1 and destination descriptors. These errors can be invalid class,
: 4340 4443 1 invalid data type, or invalid combination of a class and data type.
: 4341 4444 1 It can also tell which descriptors are supported by this
: 4342 4445 1 routine.
: 4343 4446 1
: 4344 4447 1 b. It figures out what the conversion path is; ie,
: 4345 4448 1 class, dtype --> class, dtype. These paths are given names such
: 4346 4449 1 as K_SMLINT_DEC, which reads "from small integer to decimal"
: 4347 4450 1 (categories defined later).
: 4348 4451 1
: 4349 4452 1 c. Converts the source data to an intermediate data. The strategy
: 4350 4453 1 used to select the appropriate intermediate data is explained later.
: 4351 4454 1
: 4352 4455 1 d. Puts whatever information is needed about the source and destination
: 4353 4456 1 descriptor in two structures passed by DBG$CVT_DX_DX. These
: 4354 4457 1 two structures, SRC_INFO and DST_INFO, contain the kind of
: 4355 4458 1 information that can only be visible when the class and data
: 4356 4459 1 type of the source and destination descriptors are being
: 4357 4460 1 manipulated. These two structures can be expanded to contain
: 4358 4461 1 more information as new class, and data types may require it.
: 4359 4462 1
: 4360 4463 1 This routine is comprised of a Deterministic Finite Automaton, defined
: 4361 4464 1 as a 5 tuple:
: 4362 4465 1 States : There is a state for each CLASS, and CLASS, DATA TYPE
: 4363 4466 1 combination.
: 4364 4467 1 Alphabet : Classes and Data types.
: 4365 4468 1 Mappings : M(CLASS_S , DTYPE_B) := CLASS_S_DTYPE_B
: 4366 4469 1 . . . . .
: 4367 4470 1 M(CLASS_D , DTYPE_W) := error
: 4368 4471 1 . . . . .
: 4369 4472 1 . . . . .
: 4370 4473 1 . . . . .
: 4371 4474 1 Start state :
: 4372 4475 1 Final states : All possible combinations of CLASS, DTYPE.
: 4373 4476 1 Some of these combinations are allowed, others
: 4374 4477 1 are not. The error combinations are denoted by
: 4375 4478 1 negative numbers as states.
: 4376 4479 1
: 4377 4480 1
: 4378 4481 1 MAINTENANCE OF THIS ROUTINE:
: 4379 4482 1
: 4380 4483 1 This routine knows about all classes and data types of Appendix C V8.3.
: 4381 4484 1 (You may want to update the above line everytime a change is made)
: 4382 4485 1 To make an already existing CLASS, DATA TYPE combination a valid one, as
: 4383 4486 1 opposed to an error you must:
: 4384 4487 1 1. Insert the symbol for that data type in DTYPE_TABLE in place of the
: 4385 4488 1 error state.
: 4386 4489 1 2. Define a FINAL_STATE for this combination.
: 4387 4490 1 3. Give it an action routine.
: 4388 4491 1

```



```
4389 4492 1 To add a new data type you must:
4390 4493 1 1. Increment K_MAX_DATA_TYPES.
4391 4494 1 2. Set K_MAX_DTYPE_STA to value of the new data type.
4392 4495 1 3. Does any of the following need to be changed?
4393 4496 1 a. K_SMLFINSTA
4394 4497 1 b. K_LRGFINSTA
4395 4498 1 c. K_TOP_SD
4396 4499 1 d. K_BOTTOM_SD
4397 4500 1 4. Define a new FINAL STATE.
4398 4501 1 5. Each category in DTYPE_TABLE must have a new entry for the data type.
4399 4502 1 Note that the position (starting at 0) of each entry in each category is equivalent
4400 4503 1 to the data type value.
4401 4504 1 6. Add the new label into the action routines CASE statement and
4402 4505 1 the sub-CASE statements in DBG$CVT_DX_DX will need to be modified to
4403 4506 1 include this new data type.
4404 4507 1
4405 4508 1 To add a new class you must:
4406 4509 1 1. Increment K_MAX_CLASSES
4407 4510 1 2. Set K_MAX_CLASS_STA to value of the new class.
4408 4511 1 3. Increment K_ACTUAL_CLASSES.
4409 4512 1 4. Make a new R_STATEX_CLASS_y, where x is class value and y is the
4410 4513 1 symbol of the class.
4411 4514 1 5. Make a new FINAL STATE.
4412 4515 1 6. Add a new category to the STATES structure at the end, with a index
4413 4516 1 value of one higher than the last category.
4414 4517 1 7. Make a new entry in CLASS_TABLE.
4415 4518 1 8. Make a new category in DTYPE_TABLE.
4416 4519 1 9. Make a new label in the action routine CASE statement.
4417 4520 1
4418 4521 1
4419 4522 1 CALLING SEQUENCE:
4420 4523 1 ret_status.wlc.v = FIND_CVT_PATH (SOURCE.rx.dx,
4421 4524 1 DESTINATION.rx.dx,
4422 4525 1 SRC_INFO.wr.r,
4423 4526 1 DST_INFO.wr.r,
4424 4527 1 CVT_PATH.wlu.r)
4425 4528 1
4426 4529 1 FORMAL PARAMETERS:
4427 4530 1 SOURCE Address of source descriptor passed to DBG$CVT_DX_DX.
4428 4531 1 DESTINATION Address of destination descriptor passed to DBG$CVT_DX_DX.
4429 4532 1 SRC_INFO Address of a record in DBG$CVT_DX_DX. Source information goes here.
4430 4533 1 DST_INFO Address of a record in DBG$CVT_DX_DX. Destination info goes here.
4431 4534 1 CVT_PATH Address of a longword in DBG$CVT_DX_DX. This code will determine which
4432 4535 1 CASE label is taken in DBG$CVT_DX_DX.
4433 4536 1
4434 4537 1 IMPLICIT INPUTS:
4435 4538 1 NONE
4436 4539 1
4437 4540 1 IMPLICIT OUTPUTS:
4438 4541 1 NONE
4439 4542 1
4440 4543 1 COMPLETION STATUS: (or ROUTINE VALUE:)
4441 4544 1 K_UNSCALAROU : -1 Unsupported CLASS by routine.
4442 4545 1 K_UNSDTYROU : -2 Unsupported DTYPE by routine.
4443 4546 1 K_UNSDESROU : -3 Unsupported descriptor by routine.
4444 4547 1 K_UNSDESSTA : -4 Unsupported descriptor by standard.
4445 4548 1 K_UNSCLASTA : -5 Unsupported CLASS by standard.
```

```

4446 4549 1      K_UNSDTYSTA      : -6 Unsupported DTYPE by standard.
4447 4550 1      K_INVNBDS      : -7 Invalid NBDS because array size is greater
4448 4551 1      :               than WU or dimension is not one.
4449 4552 1      K_SUPPORTED    : 1 This descriptor is supported.
4450 4553 1
4451 4554 1      SIDE EFFECTS:
4452 4555 1      Caller of DBG$CVT_DX_DX must have LIB$EMULATE as a handler, if the
4453 4556 1      source or destination descriptor explicitly ask for G, H, O conversions.
4454 4557 1
4455 4558 2      BEGIN
4456 4559 2      LOCAL
4457 4560 2      STATUS,           ! Status of this routine
4458 4561 2      STATE,           ! State
4459 4562 2      CLASS,          ! Current CLASS being looked at
4460 4563 2      DTYPE,          ! Current DTYPE being looked at
4461 4564 2      TOKEN,         ! The value of each data type supported
4462 4565 2      LEFT_CVT : VECTOR [1], ! Left side of conversion index.
4463 4566 2      RIGHT_CVT : VECTOR [1], ! Right side of conversion index.
4464 4567 2      LEFT_OR_RIGHT_CVT : REF VECTOR, ! Left or right side of conversion index.
4465 4568 2      SRC_OR_DST_INFO : REF BLOCK [, BYTE], ! Source or destination info.
4466 4569 2      SRC_OR_DST : REF BLOCK [, BYTE], ! Source or destination.
4467 4570 2      TEMP_BUF : BLOCK [K_INTMED_DATA_LENGTH, BYTE]; ! Temporary buffer for reshuffling things.
4468 4571 2
4469 4572 2      MAP
4470 4573 2      SOURCE : REF BLOCK [, BYTE],
4471 4574 2      DESTINATION : REF BLOCK [, BYTE],
4472 4575 2      SRC_INFO : REF BLOCK [, BYTE] FIELD (SRC_INFO_FIELDS),
4473 4576 2      DST_INFO : REF BLOCK [, BYTE] FIELD (DST_INFO_FIELDS);
4474 4577 2
4475 4578 2
4476 4579 2      ! Traverse through the state table twice, once for source, and once for
4477 4580 2      ! the destination descriptor. Each time through, it determines a an
4478 4581 2      ! intermediate type; ie, an intermediate type for the source and an
4479 4582 2      ! intermediate type for the destination. Eg. SMLINT or LRGFLTCMPLX.
4480 4583 2      ! The action routines also build SRC_INFO, and DST_INFO, and they
4481 4584 2      ! convert the source to its intermediate value.
4482 4585 2      ! After determining the intermediate mappings for both the source and
4483 4586 2      ! destination descriptors, a formula maps both intermediate states into
4484 4587 2      ! one final state, eg. K_SMLINT LRGFLTCMPLX. This final result is used
4485 4588 2      ! as the main CASE index in DBG$CVT_DX_DX.
4486 4589 2
4487 4590 2      ! The loop goes from 0 to 3: once for source, once for destination; if it makes
4488 4591 2      ! it to .TURN EQL 2, then it exits the loop with a successful status.
4489 4592 2      ! If the state table indicates an error (eg. invalid dtype-class combination),
4490 4593 2      ! or an error is detected in an action routine (eg. size of array cannot fit in WU),
4491 4594 2      ! then the routine exits the loop with an error code.
4492 4595 2
4493 4596 3      BEGIN
4494 4597 4      STATUS = (INCRU TURN FROM 0 TO 3 DO
4495 4598 5      BEGIN
4496 4599 5
4497 4600 5
4498 4601 5      ! Determine CLASS and DTYPE of this go around, also set up LEFT_OR_RIGHT_CVT,
4499 4602 5      ! and SRC_OR_DST, and SRC_OR_DST_INFO.
4500 4603 5      ! If this is the third time through this loop, we are finished.
4501 4604 5
4502 4605 5      CASE .TURN FROM 0 TO 2 OF

```

```

: 4503      4606 5      SET
: 4504      4607 5      [0]:
: 4505      4608 6      BEGIN
: 4506      4609 6      CLASS = .SOURCE [DSC$B_CLASS];
: 4507      4610 6      DTYPE = .SOURCE [DSC$B_DTYPE];
: 4508      4611 6      SRC_OR_DST = .SOURCE;
: 4509      4612 6      SRC_OR_DST_INFO = .SRC_INFO;
: 4510      4613 6      LEFT_OR_RIGHT_CVT = LEFT_CVT;
: 4511      4614 5      END;
: 4512      4615 5      [1]:
: 4513      4616 6      BEGIN
: 4514      4617 6      CLASS = .DESTINATION [DSC$B_CLASS];
: 4515      4618 6      DTYPE = .DESTINATION [DSC$B_DTYPE];
: 4516      4619 6      SRC_OR_DST = .DESTINATION;
: 4517      4620 6      SRC_OR_DST_INFO = .DST_INFO;
: 4518      4621 6      LEFT_OR_RIGHT_CVT = RIGHT_CVT;
: 4519      4622 5      END;
: 4520      4623 5      [2]:
: 4521      4624 5      EXITLOOP K_SUPPORTED;
: 4522      4625 5      TES;
: 4523      4626 5
: 4524      4627 5
: 4525      4628 5      ! Filter out the out-of-range CLASS and DTYPE.
: 4526      4629 5      !
: 4527      4630 5      IF .CLASS GTRU K_MAX_CLASS_STA THEN EXITLOOP K_UNSCLASTA;
: 4528      4631 5      IF .DTYPE GTRU K_MAX_DTYPE_STA THEN EXITLOOP K_UNSDTYSTA;
: 4529      4632 5
: 4530      4633 5
: 4531      4634 5      ! Crank up the finite state machine. start looking in the start state.
: 4532      4635 5      !
: 4533      4636 5      STATE = .CLASS_TABLE [.CLASS];
: 4534      4637 5
: 4535      4638 5
: 4536      4639 5      ! Action code for each state that results from the start state.
: 4537      4640 5      !
: 4538      4641 5      CASE .STATE FROM K_MSTNEGERR TO K_LRGCLSSUP OF
: 4539      4642 5      SET
: 4540      4643 5      [K_INVNBDS TO K_UNSCLAROU] :
: 4541      4644 5
: 4542      4645 5
: 4543      4646 5      ! Exit the INCR with the error resulting from the
: 4544      4647 5      ! start state.
: 4545      4648 5      !
: 4546      4649 5      EXITLOOP .STATE;
: 4547      4650 5      [K_SMLCLSSUP TO K_LRGCLSSUP] :
: 4548      4651 6      BEGIN
: 4549      4652 6
: 4550      4653 6
: 4551      4654 6      ! This is a final state, but some constants need to be
: 4552      4655 6      ! applied to it yet. This is just a data type, or a
: 4553      4656 6      ! negative number if error.
: 4554      4657 6      !
: 4555      4658 6      TOKEN = .DTYPE_TABLE [.STATE, .DTYPE];
: 4556      4659 6
: 4557      4660 6
: 4558      4661 6      ! Exit INCR with the error resulting in a final state.
: 4559      4662 6      !

```

```
: 4560      4663 6      IF .TOKEN LSS 0 THEN EXITLOOP .TOKEN;
: 4561      4664 6
: 4562      4665 6
: 4563      4666 6      ! Find the final state.
: 4564      4667 6
: 4565      4668 6      STATE = FINAL_STATE (.STATE, .TOKEN);
: 4566      4669 5      END;
: 4567      4670 5      [INRANGE, OTRANGE] :
: 4568      4671 5      $DBG_ERROR ('DBGCVTDX\FIND_CVT_PATH: invalid state');
: 4569      4672 5      TES;
: 4570      4673 5
: 4571      4674 5      ! This CASE statement contains the action code for each final state other than
: 4572      4675 5      the error states.
: 4573      4676 5      The caller of this routine has set up the pointer and length of SRC_INFO
: 4574      4677 5      to be the intermediate data area (INTMED_DATA); in the CASE below we change
: 4575      4678 5      the pointer and length if needed (e.g. any NBDS), otherwise we never
: 4576      4679 5      touch it.
: 4577      4680 5      If .TURN is 0 then we are processing the left side of the conversion, when
: 4578      4681 5      it is 1 we are processing the right side of the conversion. In other words,
: 4579      4682 5      if .TURN is 0 we are looking at the CLASS, DATA TYPE of source; if .TURN
: 4580      4683 5      is 1 we are looking at CLASS, DATA TYPE of destination.
: 4581      4684 5      These action codes determine which category (e.g. K_SMLINT or K_DEC as
: 4582      4685 5      described in DBG$CVT_DX documentation) the source or destination data type
: 4583      4686 5      falls into. They also convert the source data type to an intermediate
: 4584      4687 5      data type. For more detail refer to the functional description of
: 4585      4688 5      DBG$CVT_DX_DX.
: 4586      4689 5
: 4587      4690 5      CASE .STATE FROM K_SMLFINSTA TO K_LRGFINSTA OF
: 4588      4691 5      SET
: 4589      4692 5
: 4590      4693 5      [K_S_BU, K_SD_BU, K_UBS_BU]:
: 4591      4694 6      BEGIN
: 4592      4695 6      .LEFT OR RIGHT CVT = K_SMLINT;
: 4593      4696 6      IF .STATE EQL R_SD_BU THEN
: 4594      4697 7      BEGIN
: 4595      4698 7      SRC_OR_DST_INFO [M_SCALE] =
: 4596      4699 7      .SRC_OR_DST [DSC$B_SCALE];
: 4597      4700 7      SRC_OR_DST_INFO [M_BIN_SCALE] =
: 4598      4701 7      .SRC_OR_DST [DSC$V_FL_BINSCALE];
: 4599      4702 6      END;
: 4600      4703 6      IF .TURN EQL 0
: 4601      4704 6      THEN
: 4602      4705 6      .SRC_INFO [S_POINTER] = .BLOCK [.SOURCE [DSC$A_POINTER], 0, 0, 8, 0;.,
: 4603      4706 6      BYTE];
: 4604      4707 5      END;
: 4605      4708 5
: 4606      4709 5      [K_S_WU, K_SD_WU, K_UBS_WU]:
: 4607      4710 6      BEGIN
: 4608      4711 6      .LEFT OR RIGHT CVT = K_SMLINT;
: 4609      4712 6      IF .STATE EQL R_SD_WU THEN
: 4610      4713 7      BEGIN
: 4611      4714 7      SRC_OR_DST_INFO [M_SCALE] =
: 4612      4715 7      .SRC_OR_DST [DSC$B_SCALE];
: 4613      4716 7      SRC_OR_DST_INFO [M_BIN_SCALE] =
: 4614      4717 7      .SRC_OR_DST [DSC$V_FL_BINSCALE];
: 4615      4718 6      END;
: 4616      4719 6      IF .TURN EQL 0
```

```

4617 4720 6      THEN
4618 4721 6          .SRC_INFO [S_POINTER] = .BLOCK [.SOURCE [DSC$A_POINTER], 0, 0, 16, 0;.,
4619 4722 6              BYTE];
4620 4723 5      END;
4621 4724 5
4622 4725 5      [K_S_LU, K_SD_LU, K_UBS_LU]:
4623 4726 6      BEGIN
4624 4727 6          .LEFT OR RIGHT CVT = K_LRGINT;
4625 4728 6          IF .STATE EQL R_SD_LU THEN
4626 4729 7              BEGIN
4627 4730 7                  SRC_OR_DST_INFO [M_SCALE] =
4628 4731 7                      .SRC_OR_DST [DSC$B_SCALE];
4629 4732 7                  SRC_OR_DST_INFO [M_BIN_SCALE] =
4630 4733 7                      .SRC_OR_DST [DSC$V_FL_BINSIZE];
4631 4734 6              END;
4632 4735 6          IF .TURN EQL 0
4633 4736 6          THEN
4634 4737 6              .SRC_INFO [S_POINTER] = .BLOCK [.SOURCE [DSC$A_POINTER], 0, 0, 32, 0;.,
4635 4738 6                  BYTE];
4636 4739 5      END;
4637 4740 5
4638 4741 5      [K_S_B, K_SD_B, K_UBS_B]:
4639 4742 6      BEGIN
4640 4743 6          .LEFT OR RIGHT CVT = K_SMLINT;
4641 4744 6          IF .STATE EQL R_SD_B THEN
4642 4745 7              BEGIN
4643 4746 7                  SRC_OR_DST_INFO [M_SCALE] =
4644 4747 7                      .SRC_OR_DST [DSC$B_SCALE];
4645 4748 7                  SRC_OR_DST_INFO [M_BIN_SCALE] =
4646 4749 7                      .SRC_OR_DST [DSC$V_FL_BINSIZE];
4647 4750 6              END;
4648 4751 6          IF .TURN EQL 0
4649 4752 6          THEN
4650 4753 6              .SRC_INFO [S_POINTER] = .BLOCK [.SOURCE [DSC$A_POINTER], 0, 0, 8, 1;.,
4651 4754 6                  BYTE];
4652 4755 5      END;
4653 4756 5
4654 4757 5      [K_S_W, K_SD_W, K_UBS_W]:
4655 4758 6      BEGIN
4656 4759 6          .LEFT OR RIGHT CVT = K_SMLINT;
4657 4760 6          IF .STATE EQL R_SD_W THEN
4658 4761 7              BEGIN
4659 4762 7                  SRC_OR_DST_INFO [M_SCALE] =
4660 4763 7                      .SRC_OR_DST [DSC$B_SCALE];
4661 4764 7                  SRC_OR_DST_INFO [M_BIN_SCALE] =
4662 4765 7                      .SRC_OR_DST [DSC$V_FL_BINSIZE];
4663 4766 6              END;
4664 4767 6          IF .TURN EQL 0
4665 4768 6          THEN
4666 4769 6              .SRC_INFO [S_POINTER] = .BLOCK [.SOURCE [DSC$A_POINTER], 0, 0, 16, 1;.,
4667 4770 6                  BYTE];
4668 4771 5      END;
4669 4772 5
4670 4773 5      [K_S_L, K_SD_L, K_UBS_L]:
4671 4774 6      BEGIN
4672 4775 6          .LEFT OR RIGHT CVT = K_SMLINT;
4673 4776 6          IF .STATE EQL R_SD_L THEN

```

```

4674      4777 7      BEGIN
4675      4778 7      SRC_OR_DST_INFO [M_SCALE] =
4676      4779 7      .SRC_OR_DST [DSC$B_SCALE];
4677      4780 7      SRC_OR_DST_INFO [M_BIN_SCALE] =
4678      4781 7      .SRC_OR_DST [DSC$V_FL_BINSCALE];
4679      4782 6      END;
4680      4783 6      IF .TURN EQL 0
4681      4784 6      THEN
4682      4785 6      .SRC_INFO [S_POINTER] = .BLOCK [.SOURCE [DSC$A_POINTER], 0, 0, 32, 1;,
4683      4786 6      BYTE];
4684      4787 5      END;
4685      4788 5
4686      4789 5      [K_S_V, K_S_SV, K_S_TF, K_UBS_VU, K_UBS_SVU, K_UBS_TF]:
4687      4790 6      BEGIN
4688      4791 6      .LEFT_OR_RIGHT_CVT = K_SMLINT;
4689      4792 6      SRC_OR_DST_INFO [M_LEN] = .SRC_OR_DST [DSC$W_LENGTH];
4690      4793 6      IF .TURN EQL 0
4691      4794 6      THEN
4692      4795 7      BEGIN
4693      4796 7      LOCAL
4694      4797 7      BITPOS, SRC_PTR;
4695      4798 7
4696      4799 7      IF .SOURCE [DSC$B_CLASS] EQL DSC$K_CLASS_UBS
4697      4800 7      THEN
4698      4801 7      BITPOS = .SOURCE [DSC$L_POS]
4699      4802 7      ELSE
4700      4803 7      BITPOS = 0;
4701      4804 7      SRC_PTR = .SOURCE [DSC$A_BASE];
4702      4805 7      IF .STATE EQL K_S_SV OR .STATE EQL K_UBS_SVU
4703      4806 7      THEN
4704      4807 7      .SRC_INFO [S_POINTER] = .(.SRC_PTR) < .BITPOS, .SOURCE [DSC$W_LENGTH], 1>
4705      4808 7      ELSE
4706      4809 8      BEGIN
4707      4810 8      .SRC_INFO [S_POINTER] = .(.SRC_PTR) < .BITPOS, .SOURCE [DSC$W_LENGTH], 0>;
4708      4811 8      IF .BLOCK [.SRC_INFO [S_POINTER], 0, 31, 1, 0;, BYTE]
4709      4812 8      THEN
4710      4813 8      .LEFT_OR_RIGHT_CVT = K_LRGINT;
4711      4814 7      END;
4712      4815 6      END;
4713      4816 5      END;
4714      4817 5
4715      4818 5      [K_S_Q, K_SD_Q, K_UBS_Q, K_S_QU, K_SD_QU, K_UBS_QU]:
4716      4819 6      BEGIN
4717      4820 6      .LEFT_OR_RIGHT_CVT = K_LRGINT;
4718      4821 6      IF .STATE EQL K_SD_Q OR .STATE EQL K_SD_QU
4719      4822 6      THEN
4720      4823 7      BEGIN
4721      4824 7      SRC_OR_DST_INFO [M_SCALE] =
4722      4825 7      .SRC_OR_DST [DSC$B_SCALE];
4723      4826 7      SRC_OR_DST_INFO [M_BIN_SCALE] =
4724      4827 7      .SRC_OR_DST [DSC$V_FL_BINSCALE];
4725      4828 6      END;
4726      4829 6      IF .TURN EQL 0
4727      4830 6      THEN
4728      4831 7      BEGIN
4729      4832 7      .SRC_INFO [S_POINTER] = .BLOCK [.SOURCE [DSC$A_POINTER], 0, 0, 32, 0;, BYTE];
4730      4833 7      (.SRC_INFO [S_POINTER] + 4) = .BLOCK [.SOURCE [DSC$A_POINTER] + 4, 0, 0, 32, 0;, BYTE];

```

```

: 4731 4834 7
: 4732 4835 7
: 4733 4836 8
: 4734 4837 8
: 4735 4838 8
: 4736 4839 8
: 4737 4840 8
: 4738 4841 9
: 4739 4842 9
: 4740 4843 9
: 4741 4844 9
: 4742 4845 8
: 4743 4846 8
: 4744 4847 8
: 4745 4848 7
: 4746 4849 6
: 4747 4850 5
: 4748 4851 5
: 4749 4852 5
: 4750 4853 6
: 4751 4854 6
: 4752 4855 6
: 4753 4856 7
: 4754 4857 7
: 4755 4858 7
: 4756 4859 7
: 4757 4860 7
: 4758 4861 6
: 4759 4862 6
: 4760 4863 6
: 4761 4864 7
: 4762 4865 7
: 4763 4866 7
: 4764 4867 7
: 4765 4868 8
: 4766 4869 8
: 4767 4870 8
: 4768 4871 8
: 4769 4872 8
: 4770 4873 9
: 4771 4874 9
: 4772 4875 9
: 4773 4876 9
: 4774 4877 9
: 4775 4878 8
: 4776 4879 8
: 4777 4880 8
: 4778 4881 7
: 4779 4882 6
: 4780 4883 5
: 4781 4884 5
: 4782 4885 5
: 4783 4886 6
: 4784 4887 6
: 4785 4888 6
: 4786 4889 7
: 4787 4890 7

```

```

IF .BLOCK [.SRC_INFO [S_POINTER], 4, 31, 1, 0;, BYTE]
THEN
  BEGIN
    .SRC_INFO [S_POINTER] = ..SRC_INFO [S_POINTER] XOR %X'FFFFFFFF';
    .SRC_INFO [S_POINTER] + 4 = ..SRC_INFO [S_POINTER] + 4) XOR %X'FFFFFFFF';
    IF ..SRC_INFO [S_POINTER] EQLU K_LRGST_LU
    THEN
      BEGIN
        .SRC_INFO [S_POINTER] = 0;
        .SRC_INFO [S_POINTER] + 4 = ..(SRC_INFO [S_POINTER] + 4) + 1;
      END
    ELSE
      .SRC_INFO [S_POINTER] = ..SRC_INFO [S_POINTER] + 1;
      SRC_INFO [S_SIGN] = 1;
    END;
  END;
END;

[K_S_O, K_SD_O, K_UBS_O]:
BEGIN
  .LEFT OR RIGHT CVT = K_LRGINT;
  IF .STATE EQL R_SD_O THEN
    BEGIN
      SRC_OR_DST_INFO [M_SCALE] =
        .SRC_OR_DST [DSC$B_SCALE];
      SRC_OR_DST_INFO [M_BIN_SCALE] =
        .SRC_OR_DST [DSC$V_FL_BINS_SCALE];
    END;
  IF .TURN EQL 0
  THEN
    BEGIN
      CH$MOVE (16, .SOURCE [DSC$A_POINTER], .SRC_INFO [S_POINTER]);
      IF .BLOCK [.SRC_INFO [S_POINTER], 12, 31, 1, 0;, BYTE]
      THEN
        BEGIN
          INCR I FROM 0 TO 12 BY 4 DO
            .SRC_INFO [S_POINTER] + .I = ..(SRC_INFO [S_POINTER] + .I) XOR %X'FFFFFFFF';
            IF ..SRC_INFO [S_POINTER] EQLU K_LRGST_LU
            THEN
              BEGIN
                .SRC_INFO [S_POINTER] = 0;
                INCR I FROM 4 TO 12 BY 4 DO
                  .SRC_INFO [S_POINTER] + .I = ..(SRC_INFO [S_POINTER] + .I) + 1;
                END
              ELSE
                .SRC_INFO [S_POINTER] = ..SRC_INFO [S_POINTER] + 1;
                SRC_INFO [S_SIGN] = 1;
              END;
            END;
          END;
        END;
      END;
    END;
  [K_S_F, K_SD_F, K_UBS_F]:
  BEGIN
    .LEFT OR RIGHT CVT = K_SMLFLT_CMPLX;
    IF .STATE EQL R_SD_F THEN
      BEGIN
        SRC_OR_DST_INFO [M_SCALE] =

```

```

: 4788      4891 7
: 4789      4892 7
: 4790      4893 7
: 4791      4894 6
: 4792      4895 6
: 4793      4896 6
: 4794      4897 6
: 4795      4898 5
: 4796      4899 5
: 4797      4900 5
: 4798      4901 6
: 4799      4902 6
: 4800      4903 6
: 4801      4904 7
: 4802      4905 7
: 4803      4906 7
: 4804      4907 7
: 4805      4908 7
: 4806      4909 6
: 4807      4910 6
: 4808      4911 6
: 4809      4912 7
: 4810      4913 7
: 4811      4914 7
: 4812      4915 7
: 4813      4916 7
: 4814      4917 7
: 4815      4918 7
: 4816      4919 6
: 4817      4920 5
: 4818      4921 5
: 4819      4922 5
: 4820      4923 6
: 4821      4924 6
: 4822      4925 6
: 4823      4926 7
: 4824      4927 7
: 4825      4928 7
: 4826      4929 7
: 4827      4930 7
: 4828      4931 6
: 4829      4932 6
: 4830      4933 6
: 4831      4934 7
: 4832      4935 7
: 4833      4936 7
: 4834      4937 7
: 4835      4938 7
: 4836      4939 7
: 4837      4940 7
: 4838      4941 7
: 4839      4942 7
: 4840      4943 6
: 4841      4944 5
: 4842      4945 5
: 4843      4946 5
: 4844      4947 6

      .SRC_OR_DST [DSC$B_SCALE];
      SRC_OR_DST_INFO [M_BIN_SCALE] =
      .SRC_OR_DST[DSC$V_FL_BINSIZE];
      END;
      IF .TURN EQL 0
      THEN
      .SRC_INFO [S_POINTER] = .BLOCK [.SOURCE [DSC$A_POINTER], 0, 0, 32, 0;, BYTE];
      END;
[K_S_FC, K_SD_FC, K_UBS_FC]:
      BEGIN
      .LEFT OR RIGHT CVT = K_SMLFLT_CMPLX;
      IF .STATE EQL R_SD_FC THEN
      BEGIN
      SRC_OR_DST_INFO [M_SCALE] =
      .SRC_OR_DST [DSC$B_SCALE];
      SRC_OR_DST_INFO [M_BIN_SCALE] =
      .SRC_OR_DST[DSC$V_FL_BINSIZE];
      END;
      IF .TURN EQL 0
      THEN
      BEGIN
      .SRC_INFO [S_POINTER] = .BLOCK [.SOURCE [DSC$A_POINTER], 0, 0, 32, 0;, BYTE];

      ! Intermediate data type is double complex.
      (.SRC_INFO [S_POINTER] + 8) = .BLOCK [.SOURCE [DSC$A_POINTER] + 4, 0, 0, 32, 0;, BYTE];
      END;
      END;
[K_S_D, K_SD_D, K_UBS_D]:
      BEGIN
      .LEFT OR RIGHT CVT = K_SMLFLT_CMPLX;
      IF .STATE EQL R_SD_D THEN
      BEGIN
      SRC_OR_DST_INFO [M_SCALE] =
      .SRC_OR_DST [DSC$B_SCALE];
      SRC_OR_DST_INFO [M_BIN_SCALE] =
      .SRC_OR_DST[DSC$V_FL_BINSIZE];
      END;
      IF .TURN EQL 0
      THEN
      BEGIN

      ! The intermediate data buffer is initialized to zero, so
      ! don't have to worry about filling its binary part.
      ! (Intermediate data type is double complex).
      .SRC_INFO [S_POINTER] = .BLOCK [.SOURCE [DSC$A_POINTER], 0, 0, 32, 0;, BYTE];
      (.SRC_INFO [S_POINTER] + 4) = .BLOCK [.SOURCE [DSC$A_POINTER] + 4, 0, 0, 32, 0;, BYTE];
      END;
      END;
[K_S_DC, K_SD_DC, K_UBS_DC]:
      BEGIN

```



```

: 4845      4948 6      .LEFT OR RIGHT CVT = K SMLFLT_CMPLX;
: 4846      4949 6      IF .STATE EQL R_SD_D THEN
: 4847      4950 7          BEGIN
: 4848      4951 7              SRC_OR_DST_INFO [M SCALE] =
: 4849      4952 7                  .SRC_OR_DST [DSC$B_SCALE];
: 4850      4953 7              SRC_OR_DST_INFO [M BIN_SCALE] =
: 4851      4954 7                  .SRC_OR_DST [DSC$V_FL_BINSCALE];
: 4852      4955 6          END;
: 4853      4956 6      IF .TURN EQL 0
: 4854      4957 6      THEN
: 4855      4958 6          CH$MOVE (16, .SOURCE[DSC$A_POINTER], .SRC_INFO[S_POINTER]);
: 4856      4959 5      END;
: 4857      4960 5      [K_S_G, K_SD_G, K_UBS_G]:
: 4858      4961 5          BEGIN
: 4859      4962 6              .LEFT OR RIGHT CVT = K LRGFLT_CMPLX;
: 4860      4963 6              IF .STATE EQL R_SD_G THEN
: 4861      4964 6                  BEGIN
: 4862      4965 7                      SRC_OR_DST_INFO [M SCALE] =
: 4863      4966 7                          .SRC_OR_DST [DSC$B_SCALE];
: 4864      4967 7                      SRC_OR_DST_INFO [M BIN_SCALE] =
: 4865      4968 7                          .SRC_OR_DST [DSC$V_FL_BINSCALE];
: 4866      4969 7                      END;
: 4867      4970 6                  IF .TURN EQL 0
: 4868      4971 6                  THEN
: 4869      4972 6                      BEGIN
: 4870      4973 7                          .SRC_INFO [S_POINTER] = .BLOCK [.SOURCE [DSC$A_POINTER], 0, 0, 32, 0; .BYTE];
: 4871      4974 7                          (.SRC_INFO [S_POINTER] + 4) = .BLOCK [.SOURCE [DSC$A_POINTER] + 4, 0, 0, 32, 0; .BYTE];
: 4872      4975 7                      END;
: 4873      4976 6                  END;
: 4874      4977 5          END;
: 4875      4978 5          [K_S_GC, K_SD_GC, K_UBS_GC]:
: 4876      4979 5              BEGIN
: 4877      4980 6                  .LEFT OR RIGHT CVT = K LRGFLT_CMPLX;
: 4878      4981 6                  IF .STATE EQL R_SD_GC THEN
: 4879      4982 6                      BEGIN
: 4880      4983 7                          SRC_OR_DST_INFO [M SCALE] =
: 4881      4984 7                              .SRC_OR_DST [DSC$B_SCALE];
: 4882      4985 7                          SRC_OR_DST_INFO [M BIN_SCALE] =
: 4883      4986 7                              .SRC_OR_DST [DSC$V_FL_BINSCALE];
: 4884      4987 7                          END;
: 4885      4988 6                  IF .TURN EQL 0
: 4886      4989 6                  THEN
: 4887      4990 6                      CH$MOVE (16, .SOURCE[DSC$A_POINTER], .SRC_INFO[S_POINTER]);
: 4888      4991 6                  END;
: 4889      4992 5              END;
: 4890      4993 5              [K_S_H, K_SD_H, K_UBS_H]:
: 4891      4994 5                  BEGIN
: 4892      4995 6                      .LEFT OR RIGHT CVT = K LRGFLT_CMPLX;
: 4893      4996 6                      IF .STATE EQL R_SD_H THEN
: 4894      4997 6                          BEGIN
: 4895      4998 7                              SRC_OR_DST_INFO [M SCALE] =
: 4896      4999 7                                  .SRC_OR_DST [DSC$B_SCALE];
: 4897      5000 7                              SRC_OR_DST_INFO [M BIN_SCALE] =
: 4898      5001 7                                  .SRC_OR_DST [DSC$V_FL_BINSCALE];
: 4899      5002 7                              END;
: 4900      5003 6                      IF .TURN EQL 0 THEN CH$MOVE (16, .SOURCE [DSC$A_POINTER], .SRC_INFO [S_POINTER]);
: 4901      5004 6                      END;
```

4902	5005	5	END;
4903	5006	5	
4904	5007	5	[K_S_HC, K_SD_HC, K_UBS_HC].
4905	5008	6	BEGIN
4906	5009	6	.LEFT OR RIGHT CVT = K LRGFLT, MPLX;
4907	5010	6	IF .STATE EQL R_SD_HC THEN
4908	5011	7	BEGIN
4909	5012	7	SRC_OR_DST_INFO [M SCALE] =
4910	5013	7	.SRC_OR_DST [DSC\$B_SCALE];
4911	5014	7	SRC_OR_DST_INFO [M BIN_SCALE] =
4912	5015	7	.SRC_OR_DST [DSC\$V_FL_BINS_SCALE];
4913	5016	6	END;
4914	5017	6	IF .TURN EQL 0
4915	5018	6	THEN
4916	5019	6	CH\$MOVE (32, .SOURCE [DSC\$A_POINTER], .SRC_INFO [S_POINTER]);
4917	5020	5	END;
4918	5021	5	
4919	5022	5	[K_S_T, K_SD_T, K_UBS_T]:
4920	5023	6	BEGIN
4921	5024	6	.LEFT OR RIGHT CVT = K NBDS;
4922	5025	6	SRC_OR_DST_INFO [M_LEN] = .SRC_OR_DST [DSC\$W_LENGTH];
4923	5026	6	IF .STATE EQL K_SD_T THEN
4924	5027	7	BEGIN
4925	5028	7	SRC_OR_DST_INFO [M SCALE] =
4926	5029	7	.SRC_OR_DST [DSC\$B_SCALE];
4927	5030	7	SRC_OR_DST_INFO [M BIN_SCALE] =
4928	5031	7	.SRC_OR_DST [DSC\$V_FL_BINS_SCALE];
4929	5032	6	END;
4930	5033	6	IF .TURN EQL J
4931	5034	6	THEN
4932	5035	7	BEGIN
4933	5036	7	SRC_INFO [S_POINTER] = .SOURCE [DSC\$A_POINTER];
4934	5037	6	END;
4935	5038	5	END;
4936	5039	5	
4937	5040	5	[X_S_NU, K_SD_NU]:
4938	5041	6	BEGIN
4939	5042	6	.LEFT OR RIGHT CVT = K DEC;
4940	5043	6	IF .STATE EQL R_SD_NU THEN
4941	5044	7	BEGIN
4942	5045	7	SRC_OR_DST_INFO [M SCALE] =
4943	5046	7	.SRC_OR_DST [DSC\$B_SCALE];
4944	5047	7	SRC_OR_DST_INFO [M BIN_SCALE] =
4945	5048	7	.SRC_OR_DST [DSC\$V_FL_BINS_SCALE];
4946	5049	6	END;
4947	5050	6	IF .TURN EQL 0
4948	5051	6	THEN
4949	5052	7	BEGIN
4950	5053	7	SRC_INFO [S_LEN] = 31;
4951	5054	7	CVTTP (SOURCE [DSC\$W_LENGTH], .SOURCE [DSC\$A_POINTER], LIB\$AB_CVTTP_U,
4952	5055	7	SRC_INFO [S_LEN], .SRC_INFO [S_POINTER]);
4953	5056	6	END;
4954	5057	5	END;
4955	5058	5	
4956	5059	5	[K_S_NL, K_SD_NL]:
4957	5060	6	BEGIN
4958	5061	6	.LEFT_OR_RIGHT_CVT = K_DEC;

```

: 4959      5062  6      IF .STATE EQL K_SD_NL THEN
: 4960      5063  7      BEGIN
: 4961      5064  7          SRC_OR_DST_INFO [M_SCALE] =
: 4962      5065  7              .SRC_OR_DST [DSC$B_SCALE];
: 4963      5066  7          SRC_OR_DST_INFO [M_BIN_SCALE] =
: 4964      5067  7              .SRC_OR_DST [DSC$V_FL_BINSCALE];
: 4965      5068  6      END;
: 4966      5069  6      IF .TURN EQL 0
: 4967      5070  6      THEN
: 4968      5071  7          BEGIN
: 4969      5072  7              SRC_INFO [S_LEN] = 31;
: 4970      5073  7              CVTSP (%REF-
: 4971      5074  7                  IF .SOURCE [DSC$W_LENGTH] EQL 0 THEN 0 ELSE .SOURCE [DSC$W_LENGTH] - 1),
: 4972      5075  7                  .SOURCE [DSC$A_POINTER], SRC_INFO [S_LEN], .SRC_INFO [S_POINTER]);
: 4973      5076  6          END;
: 4974      5077  5      END;
: 4975      5078  5
: 4976      5079  5      [K_S_NLO, K_SD_NLO]:
: 4977      5080  6      BEGIN
: 4978      5081  6          .LEFT OR RIGHT CVT = K_DEC;
: 4979      5082  6          IF .STATE EQL R_SD_NLO THEN
: 4980      5083  7              BEGIN
: 4981      5084  7                  SRC_OR_DST_INFO [M_SCALE] =
: 4982      5085  7                      .SRC_OR_DST [DSC$B_SCALE];
: 4983      5086  7                  SRC_OR_DST_INFO [M_BIN_SCALE] =
: 4984      5087  7                      .SRC_OR_DST [DSC$V_FL_BINSCALE];
: 4985      5088  6              END;
: 4986      5089  6          IF .TURN EQL 0
: 4987      5090  6          THEN
: 4988      5091  7              BEGIN
: 4989      5092  7                  LOCAL
: 4990      5093  7                      LF_SIGN: REF VECTOR[, BYTE],
: 4991      5094  7                      RT_SIGN: REF VECTOR[, BYTE],
: 4992      5095  7                      ZERO_FLAG,
: 4993      5096  7                      SIGN_FLAG,
: 4994      5097  7                      PACK_ZERO: VECTOR [1];
: 4995      5098  7                      PACK_ZERO = UPLIT (%P'+0');
: 4996      5099  7                      SRC_INFO [S_LEN] = 31;
: 4997      5100  7                      CH$TRANSLATE (LIB$AB_CVT_0_U, .SOURCE [DSC$W_LENGTH], .SOURCE [DSC$A_POINTER], 0,
: 4998      5101  7                          .SOURCE [DSC$W_LENGTH], TEMP_BUF);
: 4999      5102  7                      CVTTP (SOURCE [DSC$W_LENGTH], TEMP_BUF, LIB$AB_CVTTP_U, SRC_INFO [S_LEN],
: 5000      5103  7                          .SRC_INFO [S_POINTER]);
: 5001      5104  7
: 5002      5105  7      Original code turns negative NLO type into positive NLO. If the negative
: 5003      5106  7      type comes into this piece of code without CH$TRANSLATE gives Reserved
: 5004      5107  7      Operand fault. What I did here is to move the left overpunched sign
: 5005      5108  7      to the right overpunched sign then performs the conversion. After the
: 5006      5109  7      conversion, put the sign back to where it belonged.
: 5007      5110  7
: 5008      5111  7          RT_SIGN = .SOURCE [DSC$A_POINTER] + .SOURCE [DSC$W_LENGTH] - 1;
: 5009      5112  7          LF_SIGN = .SOURCE [DSC$A_POINTER];
: 5010      5113  7          ZERO_FLAG = FALSE;
: 5011      5114  7          SELECT ONE .LF_SIGN[0] OF
: 5012      5115  7              SET
: 5013      5116  7
: 5014      5117  7              ! Positive 1 -- 9
: 5015      5118  7

```

```

5016 5119 7
5017 5120 7
5018 5121 7
5019 5122 7
5020 5123 7
5021 5124 7
5022 5125 7
5023 5126 7
5024 5127 7
5025 5128 8
5026 5129 8
5027 5130 8
5028 5131 8
5029 5132 8
5030 5133 8
5031 5134 8
5032 5135 8
5033 5136 8
5034 5137 7
5035 5138 7
5036 5139 7
5037 5140 7
5038 5141 7
5039 5142 8
5040 5143 8
5041 5144 8
5042 5145 8
5043 5146 8
5044 5147 8
5045 5148 8
5046 5149 8
5047 5150 8
5048 5151 7
5049 5152 7
5050 5153 7
5051 5154 7
5052 5155 7
5053 5156 7
5054 5157 7
5055 5158 8
5056 5159 8
5057 5160 8
5058 5161 9
5059 5162 9
5060 5163 9
5061 5164 9
5062 5165 9
5063 5166 9
5064 5167 9
5065 5168 9
5066 5169 9
5067 5170 8
5068 5171 9
5069 5172 9
5070 5173 9
5071 5174 9
5072 5175 9

```

```

[XX'41' TO XX'49']: SIGN_FLAG = TRUE;
! Negative 1 -- 9
[XX'4A' TO XX'52']: SIGN_FLAG = FALSE;
! Positive 0
[XX'7B']:
BEGIN
SIGN_FLAG = TRUE;
ZERO_FLAG = TRUE;
LF_SIGN[0] = XX'30';
IF .RT_SIGN[0] EQL XX'30'
THEN
RT_SIGN[0] = XX'7B'
ELSE
RT_SIGN[0] = .RT_SIGN[0] + XX'10';
END;
! Negative 0
[XX'7D']:
BEGIN
SIGN_FLAG = FALSE;
ZERO_FLAG = TRUE;
LF_SIGN[0] = XX'30';
IF .RT_SIGN[0] EQL XX'30'
THEN
RT_SIGN[0] = XX'7D'
ELSE
RT_SIGN[0] = .RT_SIGN[0] + XX'19';
END;
[OTHERWISE]: $DBG_ERROR('DBGCVTDX\FIND_CVT_PATH');
TES;
IF NOT .ZERO_FLAG
THEN
BEGIN
IF .SIGN_FLAG
THEN
BEGIN
LF_SIGN[0] = .LF_SIGN[0] - XX'10';
IF .RT_SIGN[0] EQL XX'30'
THEN
RT_SIGN[0] = XX'7B'
ELSE
RT_SIGN[0] = .RT_SIGN[0] + XX'10';
END
ELSE
BEGIN
LF_SIGN[0] = .LF_SIGN[0] - XX'19';
IF .RT_SIGN[0] EQL XX'30'
THEN
RT_SIGN[0] = XX'7D'

```

5073 5176 9  
5074 5177 9  
5075 5178 8  
5076 5179 8  
5077 5180 7  
5078 5181 7  
5079 5182 7  
5080 5183 7  
5081 5184 7  
5082 5185 7  
5083 5186 7  
5084 5187 7  
5085 5188 7  
5086 5189 8  
5087 5190 8  
5088 5191 8  
5089 5192 8  
5090 5193 8  
5091 5194 8  
5092 5195 8  
5093 5196 8  
5094 5197 8  
5095 5198 8  
5096 5199 8  
5097 5200 8  
5098 5201 8  
5099 5202 8  
5100 5203 8  
5101 5204 7  
5102 5205 8  
5103 5206 8  
5104 5207 8  
5105 5208 8  
5106 5209 8  
5107 5210 8  
5108 5211 8  
5109 5212 8  
5110 5213 8  
5111 5214 8  
5112 5215 8  
5113 5216 8  
5114 5217 8  
5115 5218 7  
5116 5219 7  
5117 5220 7  
5118 5221 7  
5119 5222 7  
5120 5223 7  
5121 5224 6  
5122 5225 5  
5123 5226 5  
5124 5227 5  
5125 5228 6  
5126 5229 6  
5127 5230 6  
5128 5231 7  
5129 5232 7

```

ELSE
    RT_SIGN[0] = .RT_SIGN[0] + %X'19';
END;

END;

CVTTP (SOURCE [DSC$W_LENGTH], .SOURCE [DSC$A_POINTER], LIB$AB_CVTTP_O,
        SRC_INFO[S_LEN], .SRC_INFO [S_POINTER]);

! Now put the sign back.
IF .SIGN_FLAG
THEN
    BEGIN
        IF .RT_SIGN[0] EQL %X'7B'
        THEN
            RT_SIGN[0] = %X'30'
        ELSE
            RT_SIGN[0] = .RT_SIGN[0] - %X'10';

        IF .LF_SIGN[0] EQL %X'30'
        THEN
            LF_SIGN[0] = %X'7B'
        ELSE
            LF_SIGN[0] = .LF_SIGN[0] + %X'10';

        END
    ELSE
        BEGIN
            IF .RT_SIGN[0] EQL %X'7D'
            THEN
                RT_SIGN[0] = %X'30'
            ELSE
                RT_SIGN[0] = .RT_SIGN[0] - %X'19';

            IF .LF_SIGN[0] EQL %X'30'
            THEN
                LF_SIGN[0] = %X'7D'
            ELSE
                LF_SIGN[0] = .LF_SIGN[0] + %X'19';

            END;

            IF CMPP (SRC_INFO [S_LEN], .SRC_INFO [S_POINTER], %REF (1), .PACK_ZERO) EQLU 0
            THEN
                BLOCK [.SRC_INFO [S_POINTER] + .SRC_INFO [S_LEN]/2, 0, 0, 4, 0, .BYTE] = .BLOCK [
                    .LIB$AB_CVTTP_O + .SOURCE [DSC$A_POINTER], 0, 0, 4, 0, .BYTE];

            END;
        END;

        [K_S_NR, K_SD_NR]:
        BEGIN
            LEFT OR RIGHT CVT = K_DEC;
            IF .STATE EQL R_SD_NR THEN
                BEGIN
                    SRC_OR_DST_INFO [M_SCALE] =

```

5130	5233	7	
5131	5234	7	
5132	5235	7	
5133	5236	6	
5134	5237	6	
5135	5238	6	
5136	5239	7	
5137	5240	7	
5138	5241	7	
5139	5242	7	
5140	5243	8	
5141	5244	8	
5142	5245	7	
5143	5246	7	
5144	5247	7	
5145	5248	7	
5146	5249	7	
5147	5250	6	
5148	5251	5	
5149	5252	5	
5150	5253	5	
5151	5254	6	
5152	5255	6	
5153	5256	6	
5154	5257	7	
5155	5258	7	
5156	5259	7	
5157	5260	7	
5158	5261	7	
5159	5262	6	
5160	5263	6	
5161	5264	6	
5162	5265	7	
5163	5266	7	
5164	5267	7	
5165	5268	7	
5166	5269	6	
5167	5270	5	
5168	5271	5	
5169	5272	5	
5170	5273	6	
5171	5274	6	
5172	5275	6	
5173	5276	7	
5174	5277	7	
5175	5278	7	
5176	5279	7	
5177	5280	7	
5178	5281	6	
5179	5282	6	
5180	5283	6	
5181	5284	7	
5182	5285	7	
5183	5286	7	
5184	5287	7	
5185	5288	6	
5186	5289	5	

```

        .SRC_OR_DST [DSC$B_SCALE];
SRC_OR_DST_INFO [M_BIN_SCALE] =
        .SRC_OR_DST [DSC$V_FL_BINSCALE];
END;
IF .TURN EQL 0
THEN
    BEGIN
        LOCAL
            SOU_LEN;
        SOU_LEN =
        BEGIN
            IF .SOURCE [DSC$W_LENGTH] EQL 0 THEN 0 ELSE .SOURCE [DSC$W_LENGTH] - 1
        END;
        TEMP_BUF [0, 0, 8, 0] = .BLOCK [.SOURCE [DSC$A_POINTER] + .SOU_LEN, 0, 0, 8, 0; .BYTE];
        CHSMOVE (.SOU_LEN, .SOURCE [DSC$A_POINTER], TEMP_BUF + 1);
        SRC_INFO [S_LEN] = 31;
        CVTSP (SOU_LEN, TEMP_BUF, SRC_INFO [S_LEN], .SRC_INFO [S_POINTER]);
    END;
END;

[K_S_NRO, K_SD_NRO]:
BEGIN
    .LEFT OR RIGHT CVT = K_DEC;
    IF .STATE EQL R_SD_NRO THEN
        BEGIN
            SRC_OR_DST_INFO [M_SCALE] =
                .SRC_OR_DST [DSC$B_SCALE];
            SRC_OR_DST_INFO [M_BIN_SCALE] =
                .SRC_OR_DST [DSC$V_FL_BINSCALE];
        END;
    IF .TURN EQL 0
    THEN
        BEGIN
            SRC_INFO [S_LEN] = 31;
            CVTTP (SOURCE [DSC$W_LENGTH], .SOURCE [DSC$A_POINTER], LIB$AB_CVTTP_O,
                SRC_INFO [S_LEN], .SRC_INFO [S_POINTER]);
        END;
    END;

[K_S_NZ, K_SD_NZ]:
BEGIN
    .LEFT OR RIGHT CVT = K_DEC;
    IF .STATE EQL R_SD_NZ THEN
        BEGIN
            SRC_OR_DST_INFO [M_SCALE] =
                .SRC_OR_DST [DSC$B_SCALE];
            SRC_OR_DST_INFO [M_BIN_SCALE] =
                .SRC_OR_DST [DSC$V_FL_BINSCALE];
        END;
    IF .TURN EQL 0
    THEN
        BEGIN
            SRC_INFO [S_LEN] = 31;
            CVTTP (SOURCE [DSC$W_LENGTH], .SOURCE [DSC$A_POINTER], LIB$AB_CVTTP_Z,
                SRC_INFO [S_LEN], .SRC_INFO [S_POINTER]);
        END;
    END;
END;

```

5187	5290	5
5188	5291	5
5189	5292	6
5190	5293	6
5191	5294	6
5192	5295	7
5193	5296	7
5194	5297	7
5195	5298	7
5196	5299	7
5197	5300	6
5198	5301	6
5199	5302	6
5200	5303	7
5201	5304	7
5202	5305	7
5203	5306	7
5204	5307	6
5205	5308	5
5206	5309	5
5207	5310	5
5208	5311	5
5209	5312	5
5210	5313	5
5211	5314	6
5212	5315	6
5213	5316	6
5214	5317	6
5215	5318	6
5216	5319	7
5217	5320	7
5218	5321	6
5219	5322	5
5220	5323	5
5221	5324	5
5222	5325	6
5223	5326	6
5224	5327	7
5225	5328	7
5226	5329	6
5227	5330	6
5228	5331	7
5229	5332	6
5230	5333	7
5231	5334	7
5232	5335	6
5233	5336	6
5234	5337	6
5235	5338	6
5236	5339	6
5237	5340	6
5238	5341	7
5239	5342	7
5240	5343	6
5241	5344	5
5242	5345	5
5243	5346	5

```

[K_S_P, K_SD_P]:
  BEGIN
    .LEFT OR RIGHT CVT = K DEC;
    IF .STATE EQL K_SD_P THEN
      BEGIN
        SRC_OR_DST_INFO [M_SCALE] =
          .SRC_OR_DST [DSC$B_SCALE];
        SRC_OR_DST_INFO [M_BIN_SCALE] =
          .SRC_OR_DST [DSC$V_FL_BINSCALE];
      END;
    IF .TURN EQL 0
    THEN
      BEGIN
        CVTSP (SOURCE [DSC$W_LENGTH], .SOURCE [DSC$A_POINTER], %REF (31), TEMP_BUF);
        CVTSP (%REF (31), TEMP_BUF, %REF (31), .SRC_INFO [S_POINTER]);
        SRC_INFO [S_LEN] = 31;
      END;
    END;
[K_S_ZI]:
  .LEFT_OR_RIGHT_CVT = K_NBDS;
[K_D_T]:
  BEGIN
    .LEFT OR RIGHT CVT = K_NBDS;
    SRC_OR_DST_INFO [M_LEN] = .SRC_OR_DST [DSC$W_LENGTH];
    IF .TURN EQL 0
    THEN
      BEGIN
        SRC_INFO [S_POINTER] = .SOURCE [DSC$A_POINTER];
      END;
    END;
[K_A_BU, K_A_T, K_NCA_BU, K_NCA_T]:
  BEGIN
    .LEFT OR RIGHT CVT = K_NBDS;
    IF (.SRC_OR_DST [DSC$L_ARSIZE] GTR K_LRGST_WU OR .SRC_OR_DST [DSC$B_DIMCT] NEQ 1 OR
        .SRC_OR_DST [DSC$W_LENGTH] NEQ 1)
    THEN
      EXITLOOP K_INVNBDS;
    IF (.STATE EQL K_NCA_BU OR .STATE EQL K_NCA_T)
    THEN
      BEGIN
        IF .SRC_OR_DST [DSC$L_S1] NEQ 1 THEN EXITLOOP K_INVNBDS;
      END;
      SRC_OR_DST_INFO [M_SCALE] = .SRC_OR_DST [DSC$B_SCALE];
      SRC_OR_DST_INFO [M_BIN_SCALE] = .SRC_OR_DST [DSC$V_FL_BINSCALE];
      SRC_OR_DST_INFO [M_LEN] = .SRC_OR_DST [DSC$L_ARSIZE];
      IF .TURN EQL 0
      THEN
        BEGIN
          SRC_INFO [S_POINTER] = .SOURCE [DSC$A_POINTER];
        END;
      END;
    END;
[K_VS_T, K_VS_VT]:

```

```

: 5244      5347 6      BEGIN
: 5245      5348 6      .LEFT OR RIGHT_CVT = K_NBDS;
: 5246      5349 6      IF .TORN=EQL 0
: 5247      5350 6      THEN
: 5248      5351 7      BEGIN
: 5249      5352 7      SRC_INFO [S_POINTER] = .SOURCE [DSC$A_POINTER] + 2;
: 5250      5353 7      SRC_INFO [S_LEN] = .BLOCK [.SOURCE [DSC$A_POINTER], 0, 0, 16, 0;, BYTE];
: 5251      5354 7      END
: 5252      5355 6      ELSE
: 5253      5356 6      DST_INFO [D_LEN] = .DESTINATION [DSC$W_LENGTH];
: 5254      5357 5      END;
: 5255      5358 5
: 5256      5359 5      [K_VS_AC]:
: 5257      5360 6      BEGIN
: 5258      5361 6      .LEFT OR RIGHT_CVT = K_NBDS;
: 5259      5362 6      IF .TORN=EQL 0
: 5260      5363 6      THEN
: 5261      5364 7      BEGIN
: 5262      5365 7      SRC_INFO [S_POINTER] = .SOURCE [DSC$A_POINTER] + 1;
: 5263      5366 7      SRC_INFO [S_LEN] = .BLOCK [.SOURCE [DSC$A_POINTER], 0, 0, 8, 0;, BYTE];
: 5264      5367 7      END
: 5265      5368 6      ELSE
: 5266      5369 6      DST_INFO [D_LEN] = .DESTINATION [DSC$W_LENGTH];
: 5267      5370 5      END;
: 5268      5371 5
: 5269      5372 5      [K_VS_AZ]:
: 5270      5373 6      BEGIN
: 5271      5374 6      .LEFT OR RIGHT_CVT = K_NBDS;
: 5272      5375 6      IF .TORN=EQL 0
: 5273      5376 6      THEN
: 5274      5377 7      BEGIN
: 5275      5378 7      LOCAL
: 5276      5379 7      SRC_PTR: REF VECTOR[, BYTE],
: 5277      5380 7      COUNT;
: 5278      5381 7      COUNT = 0;
: 5279      5382 7      SRC_PTR = .SOURCE[DSC$A_POINTER];
: 5280      5383 7      WHILE .SRC_PTR.COUNT NEQ 0 DO
: 5281      5384 7      COUNT = .COUNT + 1;
: 5282      5385 7      SRC_INFO[S_LEN] = .COUNT;
: 5283      5386 7      SRC_INFO [S_POINTER] = .SOURCE [DSC$A_POINTER];
: 5284      5387 7      END
: 5285      5388 6      ELSE
: 5286      5389 6      DST_INFO [D_LEN] = .DESTINATION [DSC$W_LENGTH];
: 5287      5390 5      END;
: 5288      5391 5
: 5289      5392 5      [INRANGE, OVRANGE]:
: 5290      5393 5      $DBG_ERROR ('DBGCVTDX\FIND_CVT_PATH: invalid final state');
: 5291      5394 5      TES;
: 5292      5395 5      END
: 5293      5396 4      )
: 5294      5397 2      END;
: 5295      5398 2
: 5296      5399 2
: 5297      5400 2      ! Map the left and right of the conversion, (i.e. if the conversion is
: 5298      5401 2      ! K_SMLINT_LRGFLT_CMLX, then LEFT_CVT is SMLINT and RIGHT_CVT is LRGFLT_CMLX)
: 5299      5402 2      ! into a final conversion index and return with the status of this routine.
: 5300      5403 2

```

! End of INCRU, with a EXITLOOP value.  
! End of STATUS.



				OFFC	00000	FIND_CVT_PATH:			
	5E	C0	AE	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	:	4435
		OC	AE	D4	00006	MOVAB	-64(SP), SP	:	
02	00	OC	AE	CF	00009	CLRL	TURN	:	4597
003C	0021				1\$:	CASEL	TURN, #0, #2	:	4605
			0006		2\$:	.WORD	3\$-2\$, -	:	
							4\$-2\$, -	:	
							5\$-2\$	:	
	50	04	AC	D0	00014	3\$:	MOVL SOURCE, RO	:	4609
04	AE	03	A0	9A	00018		MOVZBL 3(RO), CLASS	:	
	6E	02	A0	9A	0001D		MOVZBL 2(RO), DTYPE	:	4610
	59		50	D0	00021		MOVL RO, SRC OR DST	:	4611
	5A	OC	AC	D0	00024		MOVL SRC_INFO, SRC OR DST_INFO	:	4612
08	AE	18	AE	9E	00028		MOVAB LEFT_CVT, LEFT_OR_RIGHT_CVT	:	4613
			20	11	0002D		BRB 6\$	:	4605
	50	08	AC	D0	0002F	4\$:	MOVL DESTINATION, RO	:	4617
04	AE	03	A0	9A	00033		MOVZBL 3(RO), CLASS	:	
	6E	02	A0	9A	00038		MOVZBL 2(RO), DTYPE	:	4618
	59		50	D0	0003C		MOVL RO, SRC OR DST	:	4619
	5A	10	AC	D0	0003F		MOVL DST_INFO, SRC OR DST_INFO	:	4620
08	AE	1C	AE	9E	00043		MOVAB RIGHT_CVT, LEFT_OR_RIGHT_CVT	:	4621
			05	11	00048		BRB 6\$	:	4605
	50		01	D0	0004A	5\$:	MOVL #1, STATUS	:	4624
			6E	11	0004D		BRB 12\$	:	





0603	05C6	05A2	04EB	001A6	44\$-37\$,-
0835	07F8	07A7	0649	001AE	64\$-37\$,-
0258	0258	08B0	0872	001B6	47\$-37\$,-
053E	04FA	0403	0258	001BE	51\$-37\$,-
0258	0258	0570	0531	001C6	55\$-37\$,-
0258	0258	0258	0258	001CE	64\$-37\$,-
0353	0258	0258	0258	001D6	80\$-37\$,-
0258	0258	0258	0353	001DE	87\$-37\$,-
0258	0258	0258	0258	001E6	84\$-37\$,-
0258	0258	0258	0258	001EE	88\$-37\$,-
0258	0258	0258	0258	001F6	102\$-37\$,-
0258	0258	0258	08B6	001FE	104\$-37\$,-
0258	0258	0258	0258	00206	107\$-37\$,-
0258	0258	0258	0258	0020E	113\$-37\$,-
0258	0258	0258	0258	00216	135\$-37\$,-
0258	0258	0258	0258	0021E	140\$-37\$,-
0258	0258	0258	0258	00226	143\$-37\$,-
0258	0258	0258	0258	0022E	146\$-37\$,-
08C0	0258	0258	0258	00236	149\$-37\$,-
0258	0258	0258	0258	0023E	38\$-37\$,-
0258	0258	0258	0258	00246	38\$-37\$,-
08C0	0258	0258	0258	0024E	38\$-37\$,-
0258	0258	0258	0258	00256	71\$-37\$,-
0258	0258	0258	0258	0025E	90\$-37\$,-
0258	0258	0258	0258	00266	95\$-37\$,-
0258	0258	0258	0258	0026E	94\$-37\$,-
0258	0258	0258	0258	00276	98\$-37\$,-
0258	0258	0258	0258	0027E	38\$-37\$,-
0258	0258	0258	0258	00286	38\$-37\$,-
02A0	026F	0258	0258	0028E	38\$-37\$,-
0311	02E4	039E	02D1	00296	38\$-37\$,-
04DC	046F	039E	0346	0029E	38\$-37\$,-
05C6	05A2	04EB	04A0	002A6	38\$-37\$,-
07F8	07A7	0649	0603	002AE	38\$-37\$,-
0258	0258	0872	0835	002B6	38\$-37\$,-
04FA	0403	0258	0258	002BE	38\$-37\$,-
0258	0570	0531	053E	002C6	56\$-37\$,-
0258	0258	0258	0258	002CE	56\$-37\$,-
0258	0258	0258	0258	002D6	38\$-37\$,-
0258	0258	0258	0258	002DE	38\$-37\$,-
0258	0258	08C0	0258	002E6	38\$-37\$,-
0258	0258	0258	0258	002EE	38\$-37\$,-
0258	0258	0258	0258	002F6	38\$-37\$,-
0258	0258	08C0	0258	002FE	38\$-37\$,-
0258	0258	0258	0258	00306	38\$-37\$,-
0258	0258	0258	0258	0030E	38\$-37\$,-
0258	0258	0258	0258	00316	38\$-37\$,-
0258	0258	0258	0258	0031E	38\$-37\$,-
0258	0258	0258	0258	00326	38\$-37\$,-
0258	0258	0258	0258	0032E	38\$-37\$,-
0258	0258	0258	0258	00336	38\$-37\$,-
0258	0258	0258	0258	0033E	38\$-37\$,-
0258	0258	0258	0258	00346	38\$-37\$,-
0258	0258	0258	0258	0034E	151\$-37\$,-
0258	0258	0258	0920	00356	38\$-37\$,-
0258	0258	0258	0258	0035E	38\$-37\$,-
0258	0258	0258	0258	00366	38\$-37\$,-

.....

```

H 1
15-Sep-1984 23:57:30      VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:16:44      [DEBUG.SRC]DBGCVTDX.B32;1

```

[illegible]

```

15-Sep-1984 23:57:30 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:16:44 [DEBUG.SRC]DBGCVTDX.B32;1

```

```
VAX-11 Bliss-32 V4.0-742
[DEBUG.SRC]DBGCVTDX.B32;1
```

Page 192  
(31)

[illegible]

```

J 1
15-Sep-1984 23:57:30      VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:16:44      [DEBUG.SRC]DBGCVTDX.B32;1

```

Page 193  
(31)

38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
159\$-37\$,-  
160\$-37\$,-  
161\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
39\$-37\$,-  
41\$-37\$,-  
44\$-37\$,-  
64\$-37\$,-  
47\$-37\$,-  
51\$-37\$,-  
55\$-37\$,-  
64\$-37\$,-  
80\$-37\$,-  
87\$-37\$,-  
84\$-37\$,-  
88\$-37\$,-  
102\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
38\$-37\$,-  
71\$-37\$,-  
90\$-37\$,-  
95\$-37\$,-  
94\$-37\$,-  
98\$-37\$,-

.....





				08	BE	79	11	004A1	50\$:	BRB	61\$	4690
				00000088	8F	01	D0	004A3	51\$:	MOVL	#1, @LEFT_OR_RIGHT_CVT	4759
						56	D1	004A7		CMPL	STATE, #136	4760
						10	12	004AE		BNEQ	52\$	
				6A	08	A9	90	004B0		MOVB	8(SRC_OR_DST), (SRC_OR_DST_INFO)	4763
				01		03	EF	004B4		EXTZV	#3, #T, TO(SRC_OR_DST), R0	4765
07	50	0A	A9	01		50	F0	004BA		INSV	R0, #1, #1, 7(SRC_OR_DST_INFO)	
	AA		01			OC	AE	D5	004C0	52\$:	TSTL	4767
						65	12	004C3	53\$:	BNEQ	63\$	
				51	OC	AC	D0	004C5		MOVL	SRC_INFO, R1	4769
				50	04	AC	D0	004C9		MOVL	SOURCE, R0	
				01	04	B0	32	004CD		CVTL	@4(R0), @1(R1)	
						56	11	004D2	54\$:	BRB	63\$	4690
				08	BE	01	D0	004D4	55\$:	MOVL	#1, @LEFT_OR_RIGHT_CVT	4775
				00000089	8F	56	D1	004D8		CMPL	STATE, #137	4776
						89	11	004DF		BRB	45\$	
				08	BE	01	D0	004E1	56\$:	MOVL	#1, @LEFT_OR_RIGHT_CVT	4791
				05	AA	69	B0	004E5		MOVW	(SRC_OR_DST), 5(SRC_OR_DST_INFO)	4792
						OC	AE	D5	004E9		TSTL	4793
						67	12	004EC	57\$:	BNEQ	67\$	
				51	04	AC	D0	004EE		MOVL	SOURCE, R1	4799
				0D	03	A1	91	004F2		CMPL	3(R1), #13	
						06	12	004F6		BNEQ	58\$	
				52	08	A1	D0	004F8		MOVL	8(R1), BITPOS	4801
						02	11	004FC		BRB	59\$	
						52	D4	004FE	58\$:	CLRL	BITPOS	4803
				53	04	A1	D0	00500	59\$:	MOVL	4(R1), SRC_PTR	4804
				50	OC	AC	D0	00504		MOVL	SRC_INFO, R0	4807
				29		56	D1	00508		CMPL	STATE, #41	4805
						09	13	0050B		BEQL	60\$	
				0000012C	8F	56	D1	0050D		CMPL	STATE, #300	
						08	12	00514		BNEQ	62\$	
01	B0		63	61		52	EE	00516	60\$:	EXTV	BITPOS, (R1), (SRC_PTR), @1(R0)	4807
						71	11	0051C	61\$:	BRB	70\$	
01	B0		63	61		52	EF	0051E	62\$:	EXTZV	BITPOS, (R1), (SRC_PTR), @1(R0)	4810
						69	18	00524		BGEQ	70\$	4811
				08	BE	02	D0	00526		MOVL	#2, @LEFT_OR_RIGHT_CVT	4813
						63	11	0052A	63\$:	BRB	70\$	4793
				08	BE	02	D0	0052C	64\$:	MOVL	#2, @LEFT_OR_RIGHT_CVT	4820
				0000008A	8F	56	D1	00530		CMPL	STATE, #138	4821
						09	13	00537		BEQL	65\$	
				00000086	8F	56	D1	00539		CMPL	STATE, #134	
						10	12	00540		BNEQ	66\$	
				6A	08	A9	90	00542	65\$:	MOVB	8(SRC_OR_DST), (SRC_OR_DST_INFO)	4825
				01		03	EF	00546		EXTZV	#3, #T, TO(SRC_OR_DST), R0	4827
07	50	0A	A9	01		50	F0	0054C		INSV	R0, #1, #1, 7(SRC_OR_DST_INFO)	
	AA		01			OC	AE	D5	00552	66\$:	TSTL	4829
						5A	12	00555	67\$:	BNEQ	73\$	
				53	OC	AC	D0	00557		MOVL	SRC_INFO, R3	4832
				50	01	A3	D0	0055B		MOVL	1(R3), R0	
				51	04	AC	D0	0055F		MOVL	SOURCE, R1	
				52	04	A1	D0	00563		MOVL	4(R1), R2	
				60		62	D0	00567		MOVL	(R2), (R0)	
				51	04	A0	9E	0056A		MOVAB	4(R0), R1	4833
				61	04	A2	D0	0056E		MOVL	4(R2), (R1)	
						53	18	00572		BGEQ	74\$	4834
				60		60	D2	00574		MCOML	(R0), (R0)	4837

			FFFFFFFFFF	61 8F		61 D2 00577 MCOML (R1), (R1) : 4838
						60 D1 0057A CMPL (R0), #-1 : 4839
						06 12 00581 BNEQ 68\$ : 4842
						60 D4 00583 CLRL (R0) : 4843
						61 D6 00585 INCL (R1) : 4839
						02 11 00587 BRB 69\$ : 4846
						60 D6 00589 68\$: INCL (R0) : 4847
		07	A3			01 88 0058B 69\$: BISB2 #1, 7(R3) : 4690
						6A 11 0058F 70\$: BRB 79\$ : 4854
		08	BE			02 D0 00591 71\$: MOVL #2, @LEFT_OR_RIGHT_CVT : 4855
			0000009B	8F		56 D1 00595 CMPL STATE, #155 : 4858
						10 12 0059C BNEQ 72\$ : 4860
				6A	08	A9 90 0059E MOVB 8(SRC_OR_DST), (SRC_OR_DST_INFO) : 4862
				01		03 EF 005A2 EXTZV #3, #T, TO(SRC_OR_DST), R0 : 4865
07	50	0A	A9			50 F0 005A8 INSV R0, #1, #1, 7(SRC_OR_DST_INFO) : 4866
	AA		01			OC AE D5 005AE 72\$: TSTL TURN : 4869
						79 12 005B1 73\$: BNEQ 83\$ : 4870
				50		OC AC D0 005B3 MOVL SOURCE, R0 : 4871
				5B		OC AC D0 005B7 MOVL SRC_INFO, R11 : 4874
				58		01 AB D0 005BB MOVL 1(RT1), R8 : 4875
		68	04	B0		OF 10 28 005BF MOVCL #16, @4(RU), (R8) : 4876
						63 18 005C7 74\$: TS'B 15(R8) : 4877
						50 D4 005C9 CLRL I : 4878
						6048 9F 005CB 75\$: PUSHAB (I)[R8] : 4879
						6048 9F 005CE PUSHAB (I)[R8] : 4880
				9E		9E D2 005D1 MCOML @ (SP)+, @ (SP)+ : 4881
				04		OC F1 005D4 ACBL #12, #4, I, 75\$ : 4882
			FFFFFFFFFF	8F		68 D1 005DA CMPL (R8), #-1 : 4883
						12 12 005E1 BNEQ 77\$ : 4884
						68 D4 005E3 CLRL (R8) : 4885
				50		04 D0 005E5 MOVL #4, I : 4886
						6048 9F 005E8 76\$: PUSHAB (I)[R8] : 4887
						9E D6 005EB INCL @ (SP)+ : 4888
						OC F1 005ED ACBL #12, #4, I, 76\$ : 4889
						02 11 005F3 BRB 78\$ : 4890
						68 D6 005F5 77\$: INCL (R8) : 4891
		07	AB			01 88 005F7 78\$: BISB2 #1, 7(R11) : 4892
						68 11 005FB 79\$: BRB 86\$ : 4893
		08	BE			03 D0 005FD : MOVL #3, @LEFT_OR_RIGHT_CVT : 4894
			0000008B	8F		56 D1 006C1 CMPL STATE, #139 : 4895
						10 12 00608 BNEQ 82\$ : 4896
				6A	08	A9 90 0060A 81\$: MOVB 8(SRC_OR_DST), (SRC_OR_DST_INFO) : 4897
				01		03 EF 0060E EXTZV #3, #T, TO(SRC_OR_DST), R0 : 4898
07	50	0A	A9			50 F0 00614 INSV R0, #1, #1, 7(SRC_OR_DST_INFO) : 4899
	AA		01			OC AE D5 0061A 82\$: TSTL TURN : 4900
						49 12 0061D BNEQ 86\$ : 4901
				51		OC AC D0 0061F MOVL SRC_INFO, R1 : 4902
				50		04 AC D0 00623 MOVL SOURCE, R0 : 4903
		01	B1			04 B0 D0 00627 MOVL @4(R0), @1(R1) : 4904
						3A 11 0062C 83\$: BRB 86\$ : 4905
		08	BE			03 D0 0062E 84\$: MOVL #3, @LEFT_OR_RIGHT_CVT : 4906
			0000008D	8F		56 D1 00632 CMPL STATE, #141 : 4907
						10 12 00639 BNEQ 85\$ : 4908
				6A	08	A9 90 0063B MOVB 8(SRC_OR_DST), (SRC_OR_DST_INFO) : 4909
				01		03 EF 0063F EXTZV #3, #T, TO(SRC_OR_DST), R0 : 4910
07	50	0A	A9			50 F0 00645 INSV R0, #1, #1, 7(SRC_OR_DST_INFO) : 4911
	AA		01			

				0C	AE	D5	0064B	85\$:	TSTL	TURN	4910
					6D	12	0064E		BNEQ	93\$	
			50	0C	AC	D0	00650		MOVL	SRC_INFO, R0	4913
			51	01	A0	D0	00654		MOVL	1(R0), R1	
			50	04	AC	D0	00658		MOVL	SOURCE, R0	
			50	04	A0	D0	0065C		MOVL	4(R0), R0	
			61		60	D0	00660		MOVL	(R0), (R1)	
		08	A1	04	A0	D0	00663		MOVL	4(R0), 8(R1)	4918
					53	11	00668	86\$:	BRB	93\$	4690
		08	BE		03	D0	0066A	87\$:	MOVL	#3, @LEFT_OR_RIGHT_CVT	4924
		0000008C	8F		56	D1	0066E		CMPL	STATE, #140	4925
					1E	13	00675		BEQL	91\$	
					2C	11	00677		BRB	92\$	4932
		08	BE		03	D0	00679	88\$:	MOVL	#3, @LEFT_OR_RIGHT_CVT	4948
		0000008C	8F		56	D1	0067D		CMPL	STATE, #140	4949
					53	13	00684	89\$:	BEQL	96\$	
					61	11	00686		BRB	97\$	4956
		08	BE		04	D0	00688	90\$:	MOVL	#4, @LEFT_OR_RIGHT_CVT	4963
		0000009C	8F		56	D1	0068C		CMPL	STATE, #156	4964
					10	12	00693		BNEQ	92\$	
			6A	08	A9	90	00695	91\$:	MOVB	8(SRC_OR_DST), (SRC_OR_DST_INFO)	4967
			01		03	EF	00699		EXTZV	#3, #T, TO(SRC_OR_DST), R0	4969
07	50	0A	A9		50	F0	0069F		INSV	R0, #1, #1, 7(SRC_OR_DST_INFO)	
	AA		01		0C	AE	D5	006A5	92\$:	TSTL	TURN
					74	12	006A8		BNEQ	100\$	4971
			50	0C	AC	D0	006AA		MOVL	SRC_INFO, R0	4974
			51	01	A0	D0	006AE		MOVL	1(R0), R1	
			50	04	AC	D0	006B2		MOVL	SOURCE, R0	
			50	04	A0	D0	006B6		MOVL	4(R0), R0	
			61		60	7D	006BA		MOVQ	(R0), (R1)	
					6F	11	006BD	93\$:	BRB	101\$	4690
		08	BE		04	D0	006BF	94\$:	MOVL	#4, @LEFT_OR_RIGHT_CVT	4981
		0000009E	8F		56	D1	006C3		CMPL	STATE, #158	4982
					B8	11	006CA		BRB	89\$	
		08	BE		04	D0	006CC	95\$:	MOVL	#4, @LEFT_OR_RIGHT_CVT	4996
		0000009D	8F		56	D1	006D0		CMPL	STATE, #157	4997
					10	12	006D7		BNEQ	97\$	
			6A	08	A9	90	006D9	96\$:	MOVB	8(SRC_OR_DST), (SRC_OR_DST_INFO)	5000
			01		03	EF	006DD		EXTZV	#3, #T, TO(SRC_OR_DST), R0	5002
07	50	0A	A9		50	F0	006E3		INSV	R0, #1, #1, 7(SRC_OR_DST_INFO)	
	AA		01		0C	AE	D5	006E9	97\$:	TSTL	TURN
					40	12	006EC		BNEQ	101\$	5004
			51	04	AC	D0	006EE		MOVL	SOURCE, R1	
			50	0C	AC	D0	006F2		MOVL	SRC_INFO, R0	
		01	B0		10	28	006F6		MOVQ	#16, @4(R1), @1(R0)	
					30	11	006FC		BRB	101\$	4690
		08	BE		04	D0	006FE	98\$:	MOVL	#4, @LEFT_OR_RIGHT_CVT	5009
		0000009F	8F		56	D1	00702		CMPL	STATE, #159	5010
					10	12	00709		BNEQ	99\$	
			6A	08	A9	90	0070B		MOVB	8(SRC_OR_DST), (SRC_OR_DST_INFO)	5013
			01		03	EF	0070F		EXTZV	#3, #T, TO(SRC_OR_DST), R0	5015
07	50	0A	A9		50	F0	00715		INSV	R0, #1, #1, 7(SRC_OR_DST_INFO)	
	AA		01		0C	AE	D5	0071B	99\$:	TSTL	TURN
					6F	12	0071E	100\$:	BNEQ	106\$	5017
			51	04	AC	D0	00720		MOVL	SOURCE, R1	5019
			50	0C	AC	D0	00724		MOVL	SRC_INFO, R0	
		01	B0		20	28	00728		MOVQ	#32, @4(R1), @1(R0)	

				08	BE	5F	11	0072E	101\$:	BRB	106\$	4690	
				05	AA	06	D0	00730	102\$:	MOVL	#6, @LEFT_OR_RIGHT_CVT	5024	
				0000008F	8F	69	B0	00734		MOVW	(SRC_OR_DST); 5(SRC_OR_DST_INFO)	5025	
						56	D1	00738		CMPL	STATE, #143	5026	
						10	12	0073F		BNEQ	103\$		
					6A	08	A9	90	00741	MOVB	8(SRC_OR_DST), (SRC_OR_DST_INFO)	5029	
07	50	0A	A9	01	01	03	EF	00745		EXTZV	#3, #T, TO(SRC_OR_DST); R0	5031	
						50	F0	0074B		INSV	R0, #1, #1, 7(SRC_OR_DST_INFO)		
					08	0346	31	00751	103\$:	BRW	156\$	5033	
					08	05	D0	00754	104\$:	MOVL	#5, @LEFT_OR_RIGHT_CVT	5042	
					00000090	8F	56	D1	00758	CMPL	STATE, #144	5043	
						10	12	0075F		BNEQ	105\$		
					6A	08	A9	90	00761	MOVB	8(SRC_OR_DST), (SRC_OR_DST_INFO)	5046	
07	50	0A	A9	01	01	03	EF	00765		EXTZV	#3, #T, TO(SRC_OR_DST); R0	5048	
						50	F0	0076B		INSV	R0, #1, #1, 7(SRC_OR_DST_INFO)		
						0C	AE	D5	00771	105\$:	TSTL	TURN	5050
					50	3B	12	00774		BNEQ	109\$		
					05	0C	AC	D0	00776	MOVL	SRC_INFO, R0	5053	
					A0	1F	B0	0077A		MOVW	#31, 5(R0)		
05	A0	00000000G	00	04	B1	04	AC	D0	0077E	MOVL	SOURCE, R1	5054	
						01	61	26	00782	CVTTP	(R1), @4(R1), LIB\$AB_CVTTP_U, 5(R0), @1(R0)	5055	
							B0		0078D				
					08	43	11	0078F	106\$:	BRB	112\$	4690	
					08	05	D0	00791	107\$:	MOVL	#5, @LEFT_OR_RIGHT_CVT	5061	
					00000091	8F	56	D1	00795	CMPL	STATE, #145	5062	
						10	12	0079C		BNEQ	108\$		
					6A	08	A9	90	0079E	MOVB	8(SRC_OR_DST), (SRC_OR_DST_INFO)	5065	
07	50	0A	A9	01	01	03	EF	007A2		EXTZV	#3, #T, TO(SRC_OR_DST); R0	5067	
						50	F0	007A8		INSV	R0, #1, #1, 7(SRC_OR_DST_INFO)		
						0C	AE	D5	007AE	108\$:	TSTL	TURN	5069
					50	44	12	007B1	109\$:	BNEQ	115\$		
					05	0C	AC	D0	007B3	MOVL	SRC_INFO, R0	5072	
					A0	1F	B0	007B7		MOVW	#31, 5(R0)		
					51	04	AC	D0	007BB	MOVL	SOURCE, R1	5074	
							61	B5	007BF	TSTW	(R1)		
						04	12	007C1		BNEQ	110\$		
						52	D4	007C3		FLRL	R2		
					52	05	11	007C5		BRB	111\$		
						61	3C	007C7	110\$:	MOVZWL	(R1), R2		
01	B0	05	A0	04	B1	52	D7	007CA		DECL	R2		
						52	09	007CC	111\$:	CVTSP	R2, @4(R1), 5(R0), @1(R0)	5075	
						0347	31	007D4	112\$:	BRW	165\$	4690	
					08	05	D0	007D7	113\$:	MOVL	#5, @LEFT_OR_RIGHT_CVT	5081	
					00000092	8F	56	D1	007DB	CMPL	STATE, #146	5082	
						10	12	007E2		BNEQ	114\$		
					6A	08	A9	90	007E4	MOVB	8(SRC_OR_DST), (SRC_OR_DST_INFO)	5085	
					01	03	EF	007E8		EXTZV	#3, #T, TO(SRC_OR_DST); R0	5087	
07	50	0A	A9	01	01	50	F0	007EE		INSV	R0, #1, #1, 7(SRC_OR_DST_INFO)		
						0C	AE	D5	007F4	114\$:	TSTL	TURN	5089
							DB	12	007F7	115\$:	BNEQ	112\$	
					14	AE	9E	007F9		MOVAB	P.AKP, PACK_ZERO	5098	
					05	55	0C	AC	D0	00801	MOVL	SRC_INFO, R5	5099
						A5	1F	B0	00805	MOVW	#31, 5(R5)		
					5B	04	AC	D0	00809	MOVL	SOURCE, R11	5111	
					54	6B	3C	0080D		MOVZWL	(R11), R4		
					54	AB	C0	00810		ADDL2	4(R11), R4		
						54	D7	00814		DECL	RT_SIGN		

	58	04	AB	D0	00816	MOVL	4(R11), LF_SIGN	5112
			52	D4	0081A	CLRL	ZERO_FLAG	5113
41	8F		68	91	0081C	CMPB	(LF_SIGN), #65	5119
			0C	1F	00820	BLSSU	116\$	
49	8F		68	91	00822	CMPB	(LF_SIGN), #73	
			06	1A	00826	BGTRU	116\$	
10	AE		01	D0	00828	MOVL	#1, SIGN_FLAG	
			65	11	0082C	BRB	122\$	
4A	8F		68	91	0082E	CMPB	(LF_SIGN), #74	5123
			0B	1F	00832	BLSSU	117\$	
52	8F		68	91	00834	CMPB	(LF_SIGN), #82	
			05	1A	00838	BGTRU	117\$	
		10	AE	D4	0083A	CLRL	SIGN_FLAG	
			54	11	0083D	BRB	122\$	
7B	8F		68	91	0083F	CMPB	(LF_SIGN), #123	5127
			1A	12	00843	BNEQ	119\$	
10	AE		01	D0	00845	MOVL	#1, SIGN_FLAG	5129
	52		01	D0	00849	MOVL	#1, ZERO_FLAG	5130
	68		30	90	0084C	MOVB	#48, (LF_SIGN)	5131
	30		64	91	0084F	CMPB	(RT_SIGN), #48	5132
			06	12	00852	BNEQ	118\$	
	64	7B	8F	90	00854	MOVB	#123, (RT_SIGN)	5134
			39	11	00858	BRB	122\$	
	64		10	80	0085A	ADDB2	#16, (RT_SIGN)	5136
			34	11	0085D	BRB	122\$	5114
7D	8F		68	91	0085F	CMPB	(LF_SIGN), #125	5141
			19	12	00863	BNEQ	121\$	
		10	AE	D4	00865	CLRL	SIGN_FLAG	5143
	52		01	D0	00868	MOVL	#1, ZERO_FLAG	5144
	68		30	90	0086B	MOVB	#48, (LF_SIGN)	5145
	30		64	91	0086E	CMPB	(RT_SIGN), #48	5146
			06	12	00871	BNEQ	120\$	
	64	7D	8F	90	00873	MOVB	#125, (RT_SIGN)	5148
			1A	11	00877	BRB	122\$	
	64		19	80	00879	ADDB2	#25, (RT_SIGN)	5150
			15	11	0087C	BRB	122\$	5114
		00000000	EF	9F	0087E	PUSHAB	P.AKQ	5153
			01	DD	00884	PUSHL	#1	
		00028362	8F	DD	00886	PUSHL	#164706	
00000000G	00		03	FB	0088C	CALLS	#3, LIB\$SIGNAL	
	28		52	E8	00893	BLBS	ZERO_FLAG, 126\$	5156
	13	10	AE	E9	00896	BLBC	SIGN_FLAG, 124\$	5159
	68		10	82	0089A	SUBB2	#16, (LF_SIGN)	5162
	30		64	91	0089D	CMPB	(RT_SIGN), #48	5163
			06	12	008A0	BNEQ	123\$	
	64	7B	8F	90	008A2	MOVB	#123, (RT_SIGN)	5165
			16	11	008A6	BRB	126\$	
	64		10	80	008A8	ADDB2	#16, (RT_SIGN)	5167
			11	11	008AB	BRB	126\$	5159
	68		19	82	008AD	SUBB2	#25, (LF_SIGN)	5172
	30		64	91	008B0	CMPB	(RT_SIGN), #48	5173
			06	12	008B3	BNEQ	125\$	
	64	7D	8F	90	008B5	MOVB	#125, (RT_SIGN)	5175
			03	11	008B9	BRB	126\$	
	64		19	80	008BB	ADDB2	#25, (RT_SIGN)	5177
05	A5	00000000G	00	04	BB	126\$:	CVTTP	5183
			01	B5	008C9		a1(R5)	
							a4(R11), LIB\$AB_CVTTP_0, 5(R5), -	

				1E	10	AE	E9	008CB	BLBC	SIGN FLAG, 130\$	5187
				7B 8F		64	91	008CF	CMPB	(RT_SIGN), #123	5190
						05	12	008D3	BNEQ	127\$	
				64		30	90	008D5	MOVB	#48, (RT_SIGN)	5192
						03	11	008D8	BRB	128\$	
				64		10	82	008DA	SUBB2	#16, (RT_SIGN)	5194
				30		68	91	008DD	CMPB	(LF_SIGN), #48	5196
						06	12	008E0	BNEQ	129\$	
				68	7B	8F	90	008E2	MOVB	#123, (LF_SIGN)	5198
						21	11	008E6	BRB	134\$	
				68		10	80	008E8	ADDB2	#16, (LF_SIGN)	5200
						1C	11	008EB	BRB	134\$	5187
				7D 8F		64	91	008ED	CMPB	(RT_SIGN), #125	5206
						05	12	008F1	BNEQ	131\$	
				64		30	90	008F3	MOVB	#48, (RT_SIGN)	5208
						03	11	008F6	BRB	132\$	
				64		19	82	008F8	SUBB2	#25, (RT_SIGN)	5210
				30		68	91	008FB	CMPB	(LF_SIGN), #48	5212
						06	12	008FE	BNEQ	133\$	
				68	7D	8F	90	00900	MOVB	#125, (LF_SIGN)	5214
						03	11	00904	BRB	134\$	
				68		19	80	00906	ADDB2	#25, (LF_SIGN)	5216
14	BE	01	01	B5	05	A5	37	00909	CMPP4	5(R5), @T(R5), #1, @PACK_ZERO	5220
	54					54	DC	00911	MOVPSL	R4	
				02		02	EF	00913	EXTZV	#2, #2, R4, R4	
						54	D7	00918	DECL	R4	
						68	12	0091A	BNEQ	139\$	
				50	05	A5	3C	0091C	MOVZWL	5(R5), R0	5222
				50		02	C6	00920	DIVL2	#2, R0	
01	B540	51	00000000G	00	04	AB	C1	00923	ADDL3	4(R11), LIB\$AB_CVTTP_0, R1	5223
		04		00		61	F0	0092C	INSV	(R1), #0, #4, @1(R5)[R0]	5222
						4F	11	00933	BRB	139\$	4690
			08	BE		05	D0	00935	MOVL	#5, @LEFT_OR_RIGHT_CVT	5229
			00000093	8F		56	D1	00939	CMPL	STATE, #147	5230
						10	12	00940	BNEQ	136\$	
				6A	08	A9	90	00942	MOVB	8(SRC_OR_DST), (SRC_OR_DST_INFO)	5233
				01		03	EF	00946	EXTZV	#3, #T, TO(SRC_OR_DST), R0	5235
07	50	0A	A9	01		50	F0	0094C	INSV	R0, #1, #1, 7(SRC_OR_DST_INFO)	
AA			01		0C	AE	D5	00952	TSTL	TURN	5237
						6A	12	00955	BNEQ	142\$	
				50	04	AC	D0	00957	MOVL	SOURCE, R0	5244
						60	B5	0095B	TSTW	(R0)	
						04	12	0095D	BNEQ	137\$	
						5B	D4	0095F	CLRL	SOU_LEN	
						05	11	00961	BRB	138\$	
				5B		60	3C	00963	MOVZWL	(R0), SOU_LEN	
						5B	D7	00966	DECL	SOU_LEN	
			20	AE	04	B0	90	00968	MOVB	@4(R0)[SOU_LEN], TEMP_BUF	5246
			04	B0		5B	28	0096E	MOVC3	SOU_LEN, @4(R0), TEMP_BUF+1	5247
				50	0C	AC	D0	00974	MOVL	SRC_INFO, R0	5248
				A0		1F	B0	00978	MOVW	#31, 5(R0)	
01	B0	05	A0	20		5B	09	0097C	CVTSP	SOU_LEN, TEMP_BUF, 5(R0), @1(R0)	5249
						78	11	00984	BRB	145\$	4690
				08		05	D0	00986	MOVL	#5, @LEFT_OR_RIGHT_CVT	5255
			00000094	BE		56	D1	0098A	CMPL	STATE, #148	5256
				8F		10	12	00991	BNEQ	141\$	
						6A	08	00993	MOVB	8(SRC_OR_DST), (SRC_OR_DST_INFO)	5259

PC	Op	OpC	OpD	OpI	OpR	OpS	OpT	OpU	OpV	OpW	OpX	OpY	OpZ	OpAA	OpAB	OpAC	OpAD	OpAE	OpAF	OpAG	OpAH	OpAI	OpAJ	OpAK	OpAL	OpAM	OpAN	OpAO	OpAP	OpAQ	OpAR	OpAS	OpAT	OpAU	OpAV	OpAW	OpAX	OpAY	OpAZ	OpBA	OpBB	OpBC	OpBD	OpBE	OpBF	OpBG	OpBH	OpBI	OpBJ	OpBK	OpBL	OpBM	OpBN	OpBO	OpBP	OpBQ	OpBR	OpBS	OpBT	OpBU	OpBV	OpBW	OpBX	OpBY	OpBZ	OpCA	OpCB	OpCC	OpCD	OpCE	OpCF	OpCG	OpCH	OpCI	OpCJ	OpCK	OpCL	OpCM	OpCN	OpCO	OpCP	OpCQ	OpCR	OpCS	OpCT	OpCU	OpCV	OpCW	OpCX	OpCY	OpCZ	OpDA	OpDB	OpDC	OpDD	OpDE	OpDF	OpDG	OpDH	OpDI	OpDJ	OpDK	OpDL	OpDM	OpDN	OpDO	OpDP	OpDQ	OpDR	OpDS	OpDT	OpDU	OpDV	OpDW	OpDX	OpDY	OpDZ	OpEA	OpEB	OpEC	OpED	OpEE	OpEF	OpEG	OpEH	OpEI	OpEJ	OpEK	OpEL	OpEM	OpEN	OpEO	OpEP	OpEQ	OpER	OpES	OpET	OpEU	OpEV	OpEW	OpEX	OpEY	OpEZ	OpFA	OpFB	OpFC	OpFD	OpFE	OpFF	OpFG	OpFH	OpFI	OpFJ	OpFK	OpFL	OpFM	OpFN	OpFO	OpFP	OpFQ	OpFR	OpFS	OpFT	OpFU	OpFV	OpFW	OpFX	OpFY	OpFZ	OpGA	OpGB	OpGC	OpGD	OpGE	OpGF	OpGG	OpGH	OpGI	OpGJ	OpGK	OpGL	OpGM	OpGN	OpGO	OpGP	OpGQ	OpGR	OpGS	OpGT	OpGU	OpGV	OpGW	OpGX	OpGY	OpGZ	OpHA	OpHB	OpHC	OpHD	OpHE	OpHF	OpHG	OpHH	OpHI	OpHJ	OpHK	OpHL	OpHM	OpHN	OpHO	OpHP	OpHQ	OpHR	OpHS	OpHT	OpHU	OpHV	OpHW	OpHX	OpHY	OpHZ	OpIA	OpIB	OpIC	OpID	OpIE	OpIF	OpIG	OpIH	OpII	OpIJ	OpIK	OpIL	OpIM	OpIN	OpIO	OpIP	OpIQ	OpIR	OpIS	OpIT	OpIU	OpIV	OpIW	OpIX	OpIY	OpIZ	OpJA	OpJB	OpJC	OpJD	OpJE	OpJF	OpJG	OpJH	OpJI	OpJJ	OpJK	OpJL	OpJM	OpJN	OpJO	OpJP	OpJQ	OpJR	OpJS	OpJT	OpJU	OpJV	OpJW	OpJX	OpJY	OpJZ	OpKA	OpKB	OpKC	OpKD	OpKE	OpKF	OpKG	OpKH	OpKI	OpKJ	OpKK	OpKL	OpKM	OpKN	OpKO	OpKP	OpKQ	OpKR	OpKS	OpKT	OpKU	OpKV	OpKW	OpKX	OpKY	OpKZ	OpLA	OpLB	OpLC	OpLD	OpLE	OpLF	OpLG	OpLH	OpLI	OpLJ	OpLK	OpLL	OpLM	OpLN	OpLO	OpLP	OpLQ	OpLR	OpLS	OpLT	OpLU	OpLV	OpLW	OpLX	OpLY	OpLZ	OpMA	OpMB	OpMC	OpMD	OpME	OpMF	OpMG	OpMH	OpMI	OpMJ	OpMK	OpML	OpMM	OpMN	OpMO	OpMP	OpMQ	OpMR	OpMS	OpMT	OpMU	OpMV	OpMW	OpMX	OpMY	OpMZ	OpNA	OpNB	OpNC	OpND	OpNE	OpNF	OpNG	OpNH	OpNI	OpNJ	OpNK	OpNL	OpNM	OpNN	OpNO	OpNP	OpNQ	OpNR	OpNS	OpNT	OpNU	OpNV	OpNW	OpNX	OpNY	OpNZ	OpOA	OpOB	OpOC	OpOD	OpOE	OpOF	OpOG	OpOH	OpOI	OpOJ	OpOK	OpOL	OpOM	OpON	OpOO	OpOP	OpOQ	OpOR	OpOS	OpOT	OpOU	OpOV	OpOW	OpOX	OpOY	OpOZ	OpPA	OpPB	OpPC	OpPD	OpPE	OpPF	OpPG	OpPH	OpPI	OpPJ	OpPK	OpPL	OpPM	OpPN	OpPO	OpPP	OpPQ	OpPR	OpPS	OpPT	OpPU	OpPV	OpPW	OpPX	OpPY	OpPZ	OpQA	OpQB	OpQC	OpQD	OpQE	OpQF	OpQG	OpQH	OpQI	OpQJ	OpQK	OpQL	OpQM	OpQN	OpQO	OpQP	OpQQ	OpQR	OpQS	OpQT	OpQU	OpQV	OpQW	OpQX	OpQY	OpQZ	OpRA	OpRB	OpRC	OpRD	OpRE	OpRF	OpRG	OpRH	OpRI	OpRJ	OpRK	OpRL	OpRM	OpRN	OpRO	OpRP	OpRQ	OpRR	OpRS	OpRT	OpRU	OpRV	OpRW	OpRX	OpRY	OpRZ	OpSA	OpSB	OpSC	OpSD	OpSE	OpSF	OpSG	OpSH	OpSI	OpSJ	OpSK	OpSL	OpSM	OpSN	OpSO	OpSP	OpSQ	OpSR	OpSS	OpST	OpSU	OpSV
----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------



07	50 AA	0A	A9 01	05	6A 01 01 AA	08	00A8 A9 03 50 A9 AE 7F	31 90 EF F0 B0 D5 12	00A82 00A85 00A89 00A8F 00A95 00A9A 00A9D	155\$:     156\$: 157\$:	BRW MOVB EXTZV INSV MOVW TSTL BNEQ	167\$ 8(SRC OR DST), (SRC OR DST_INFO) #3, #T, TO(SRC OR DST), RO RO, #1, #1, 7(SRC OR DST_INFO) 12(SRC OR DST), 5(SRC OR DST_INFO) TURN 165\$	5336 5337  5338 5339
					51 50 A1	0C 04 04	AC AC A0	D0 D0 D0	00A9F 00AA3 00AA7		MOVL MOVL MOVL	SRC_INFO, R1 SOURCE, RO 4(RO), 1(R1)	5342
				08	BE		70 06 AE 5E	11 D0 D5 12	00AAC 00AAE 00AB2 00AB5	158\$: 159\$:	BRB MOVL TSTL BNEQ	165\$ #6, @LEFT_OR_RIGHT_CVT TURN 164\$	4690 5348 5349
		01	A0		50 51 A1 A0	0C 04 04	AC AC 02	D0 D0 C1	00AB7 00ABB 00ABF		MOVL MOVL ADDL3	SRC_INFO, RO SOURCE, R1 #2, 4(R1), 1(RO)	5352
				05	A0	04	B1 52 06	B0 11 D0	00AC5 00ACA 00ACC		MOVW BRB MOVL	@4(R1), 5(RO) 165\$ #6, @LEFT_OR_RIGHT_CVT	5353 5349 5361
				08	BE		AE 40 12	D5 00AD3 00AD5	00AD0 00AD3 00AD5	160\$:	TSTL BNEQ MOVL	TURN 164\$ SRC_INFO, RO	5362
		01	A0		50 51 A1 A0	0C 04 04	AC AC 01	D0 D0 C1	00AD9 00ADD 00ADD		MOVL MOVL ADDL3	SOURCE, R1 #1, 4(R1), 1(RO)	5365
				05	A0	04	B1 34 11	9B 00AE3 00AEB	00AE3 00AE3 00AEB		MOVZBW BRB MOVL	@4(R1), 5(RO) 165\$ #6, @LEFT_OR_RIGHT_CVT	5366 5362 5374
				08	BE		06 AE D5	D0 00AEA 00AEE	00AEA 00AEA 00AEE	161\$:	TSTL BNEQ CLRL	TURN 164\$ COUNT	5375
					51 50	04 04	AC A1	D0 D0	00AF3 00AF5 00AF9		MOVL MOVL MOVL	SOURCE, R1 4(R1), SRC_PTR (COUNT)[SRC_PTR]	5381 5382
							6240 04 52	95 13 D6	00AFD 00B00 00B02	162\$:	TSTB BEQL INCL	163\$ COUNT 162\$	5383
					50	0C	AC	D0	00B06	163\$:	BRB MOVL	162\$ SRC_INFO, RO	5384 5385
		05	A0		A0	04	52 A1	B0 D0	00B0A 00B0E		MOVW MOVL	COUNT, 5(RO) 4(R1), 1(RO)	5386
					50	09	11	00B13	00B13		BRB	165\$	5375
		05	A0		A0	10	AC	D0	00B15	164\$:	MOVL	DST_INFO, RO	5389
						08	BC	B0	00B19		MOVW	@DESTINATION, 5(RO)	
					03	0C	AE	D6	00B1E	165\$:	INCL	TURN	4597
						0C	AE	D1	00B21		CMPL	TURN, #3	
							03	1A	00B25		BGTRU	166\$	
					50		F4DF 01	31 CE	00B27 00B2A	166\$:	BRW MNEGL	1\$ #1, STATUS	
		51		18	AE		06	C5	00B2D	167\$:	MULL3	#6, LEFT_CVT, R1	5404
					51	1C	AE	C0	00B32		ADDL2	RIGHT_CVT, R1	
				14	BC	FA	A1	9E	00B36		MOVAB	-6(R1), @CVT_PATH	
								04	00B3B		RET		5406

; Routine Size: 2876 bytes. Routine Base: DBG\$CODE + 347F

; 5304 5407 1

DBGCVTDX  
V04-000

H 2  
15-Sep-1984 23:57:30 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:16:44 [DEBUG.SRC]DBGCVTDX.B32;1

Page 204  
(31)

: 5305 5408 1 END  
: 5306 5409 0 ELUDOM

! End of module DBGCVTDX.

.EXTRN LIB\$SIGNAL, SYSSUNWIND

# PSECT SUMMARY

Name	Bytes	Attributes
DBG\$OWN	208	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
DBG\$PLIT	6740	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(0)
DBG\$CODE	16315	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(0)

# Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	59	0	1000	00:01.8
_\$255\$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1	32	0	0	7	00:00.1
_\$255\$DUA28:[DEBUG.OBJ]DBGLIB.L32;1	1545	101	6	97	00:02.0
_\$255\$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1	418	6	1	31	00:00.3
_\$255\$DUA28:[DEBUG.OBJ]DBGMSG.L32;1	386	18	4	22	00:00.3

: Information: 2  
: Warnings: 0  
: Errors: 0

# COMMAND QUALIFIERS

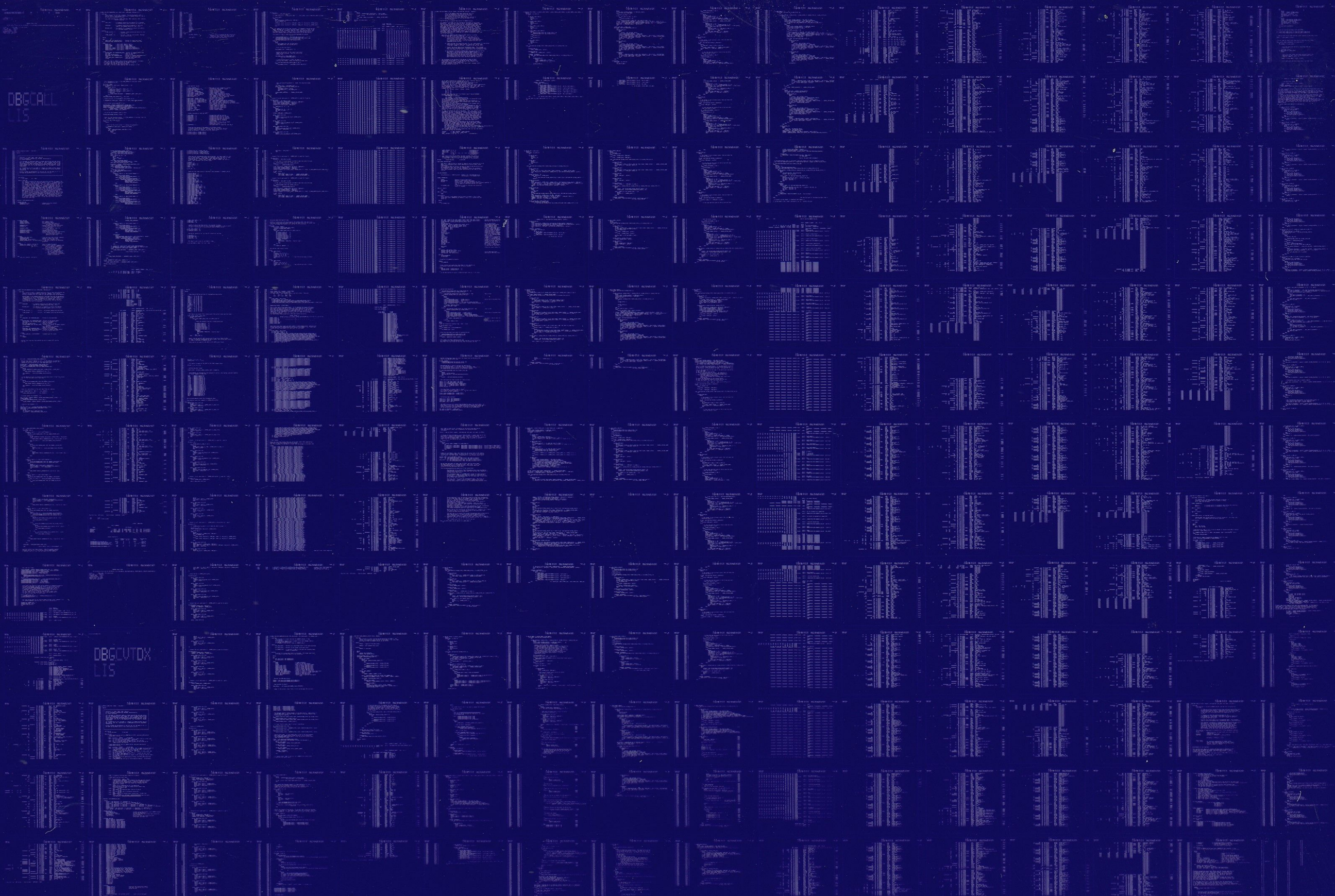
: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:DBGCVTDX/OBJ=OBJ\$:DBGCVTDX MSRC\$:DBGCVTDX/UPDATE=(ENH\$:DBGCVTDX)

: Size: 16315 code + 6948 data bytes  
: Run Time: 05:24.7  
: Elapsed Time: 17:37.9  
: Lines/CPU Min: 999  
: Lexemes/CPU-Min: 14517  
: Memory Used: 2755 pages  
: Compilation Complete



0078 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY





0079 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

